Numerical Simulation and Scientific Computing I

Lecture 10: Mesh Generation

Clemens Etl, Paul Manstetten, and Josef Weinbub

Institute for Microelectronics
TU Wien

nssc@iue.tuwien.ac.at

WS 2023

Outline

- **Quiz Wrapup**
- Mesh Generation
- Next Quiz

Quiz – Q1 – Poll 1

- Q1: What happens if the number of threads and the number of SECTIONs are different? More threads than SECTIONs? Less threads than SECTIONs?
- A) More: some threads stall, less: implementation-defined
- B) More: some SECTIONs repeat, less: some SECTIONs will not run
- C) More: some threads stall, less: some SECTIONs will not run
- D) some SECTIONs repeat, less: implementation-defined

Quiz – Q1 – Poll 1

- Q1: What happens if the number of threads and the number of SECTIONs are different? More threads than SECTIONs? Less threads than SECTIONs?

- **A) More: some threads stall, less: implementation-defined**

- B) More: some SECTIONs repeat, less: some SECTIONs will not run

- C) More: some threads stall, less: some SECTIONs will not run

- D) some SECTIONs repeat, less: implementation-defined

Quiz – Q2 – Poll 2

- Q2: Consider the following code: Which loops will be collapsed? Which loop iteration variables must be made private?

```
#pragma omp for collapse(3)
for (i=0; i<imax; i++)
  for (j=0; j<jmax; j++)
    for (k=0; k<kmax; k++)
       for (l=0; l<lmax; l++)
           for (m=0; m<mmax; m++)
```

- A) collapse: i,j,k; make private: i,j,k
- B) collapse: i,j,k; make private: l,m
- C) collapse: k,l,m; make private: k,l,m
- D) collapse: k,l,m; make private: i,j

Quiz – Q2 – Poll 2

- Q2: Consider the following code: Which loops will be collapsed? Which loop iteration variables must be made private?

```
#pragma omp for collapse(3)
for (i=0; i<imax; i++)
  for (j=0; j<jmax; j++)
    for (k=0; k<kmax; k++)
       for (l=0; l<lmax; l++)
            for (m=0; m<mmax; m++)
```

- A) collapse: i,j,k; make private: i,j,k
- **B) collapse: i,j,k; make private: l,m**
- C) collapse: k,l,m; make private: k,l,m
- D) collapse: k,l,m; make private: i,j

Quiz – Q3

- Q3: Consider the following code and substituting `XXX` with `private`, `firstprivate`, and `lastprivate`: What is the state of `x` before `x=i`? What will the `cout` statement output and why?

```
#include <iostream>
#include <omp.h>
main(){
  int i, x=44;
  #pragma omp parallel for XXX(x)
  for(i=0;i<=10;i++)
    x=i;
  std::cout << "final x: " << x << std::endl;
}
```

Quiz – Q3 – Polls 3&4

```
#include <iostream>
#include <omp.h>
main(){
  int i, x=44;
  #pragma omp parallel for private(x)
  for(i=0;i<=10;i++)
    x=i;
  std::cout << "final x: " << x << std::endl;
}
```

- 3) What is the state of `x` before `x=i`?
- A) 10
- B) 44
- C) Undefined

- 4) What will the `cout` statement output?
- A) 10
- B) 44
- C) Undefined

Quiz – Q3 – Polls 3&4

```
#include <iostream>
#include <omp.h>
main(){
  int i, x=44;
  #pragma omp parallel for private(x)
  for(i=0;i<=10;i++)
    x=i;
  std::cout << "final x: " << x << std::endl;
}
```

- 3) What is the state of `x` before `x=i`?

- A) 10
- B) 44
- **C) Undefined**

- 4) What will the `cout` statement output?

- A) 10
- **B) 44**
- C) Undefined

Quiz – Q3 – Polls 5&6

```
#include <iostream>
#include <omp.h>
main(){
  int i, x=44;
  #pragma omp parallel for firstprivate(x)
  for(i=0;i<=10;i++)
    x=i;
  std::cout << "final x: " << x << std::endl;
}
```

5) What is the state of `x` before `x=i`?

- A) 10
- B) 44
- C) Undefined

- 6) What will the `cout` statement output?

- A) 10
- B) 44
- C) Undefined

Quiz – Q3 – Polls 5&6

```
#include <iostream>
#include <omp.h>
main(){
  int i, x=44;
  #pragma omp parallel for firstprivate(x)
  for(i=0;i<=10;i++)
    x=i;
  std::cout << "final x: " << x << std::endl;
}
```

5) What is the state of `x` before `x=i`?

- A) 10
- **B) 44**
- C) Undefined

- 6) What will the `cout` statement output?

- A) 10
- **B) 44**
- C) Undefined

Quiz – Q3 – Polls 7&8

```
#include <iostream>
#include <omp.h>
main(){
  int i, x=44;
  #pragma omp parallel for lastprivate(x)
  for(i=0;i<=10;i++)
    x=i;
  std::cout << "final x: " << x << std::endl;
}
```

7) What is the state of `x` before `x=i`?

- A) 10
- B) 44
- C) Undefined

- 8) What will the `cout` statement output?

- A) 10
- B) 44
- C) Undefined

Quiz – Q3 – Polls 7&8

```
#include <iostream>
#include <omp.h>
main(){
  int i, x=44;
  #pragma omp parallel for lastprivate(x)
  for(i=0;i<=10;i++)
    x=i;
  std::cout << "final x: " << x << std::endl;
}
```

7) What is the state of `x` before `x=i`?

- A) 10
- B) 44
- **C) Undefined**

- 8) What will the `cout` statement output?

- **A) 10**
- B) 44
- C) Undefined

Outline

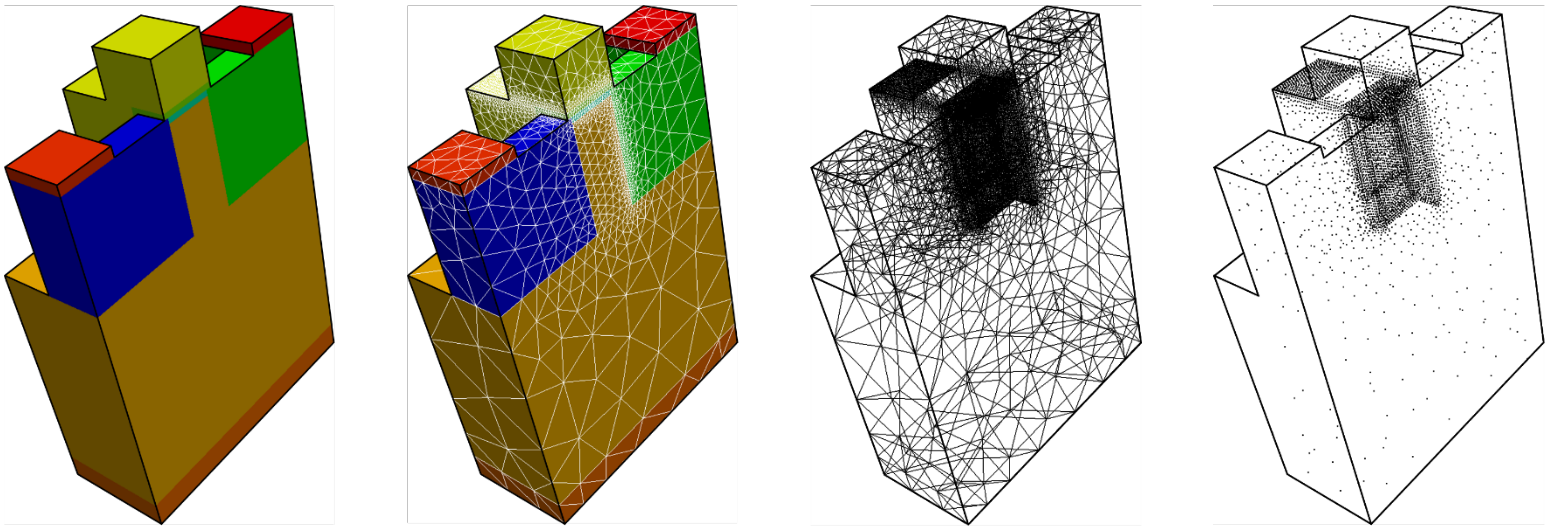
- Quiz Wrapup
- **Mesh Generation**
- Next Quiz

Sources

- Florian Rudolf, Dissertation TU Wien
  Symmetry- and Similarity-Aware Volumetric Meshing
  http://www.iue.tuwien.ac.at/phd/rudolf/
- René Heinzl, Dissertation TU Wien
  Concepts for Scientific Computing
  http://www.iue.tuwien.ac.at/phd/heinzl/
- Marshall Bern and Paul Plassmann
  Mesh Generation
  https://www.ics.uci.edu/~eppstein/280g/Bern-Plassman-meshgen.pdf
- Marshall Bern and David Eppstein
  Mesh Generation and Optimal Triangulation
  https://www.ics.uci.edu/~eppstein/pubs/BerEpp-CEG-95.pdf
- Paolo Cignoni
  Data Structures for 3D Meshes
  http://vcg.isti.cnr.it/~cignoni/SciViz1920/

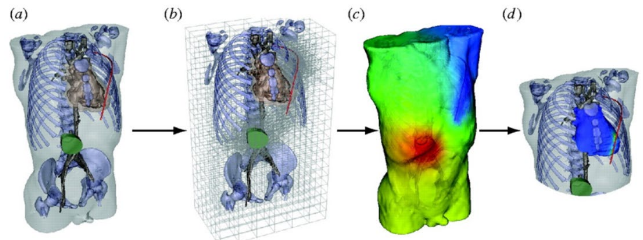
What is Meshing?

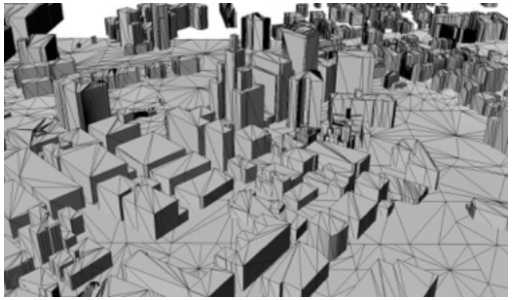
Meshing is the <u>discretization</u> of a continuous simulation domain, represented by a <u>geometry</u>, into a discrete <u>mesh</u> to obtain a finite representation.

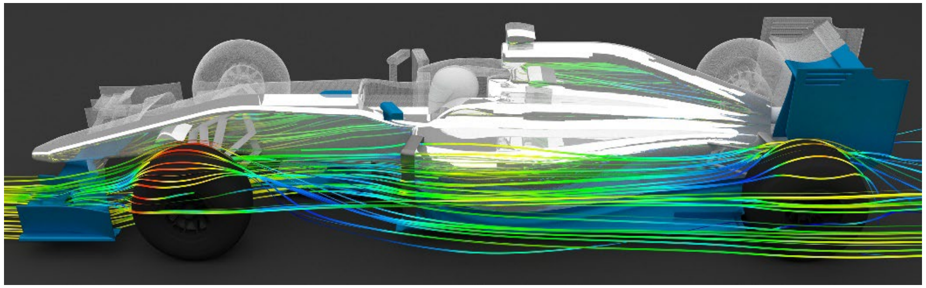


Why are Meshes Important?

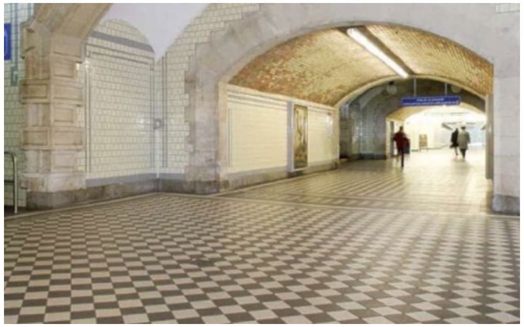
- **Central aspect:**
  Solution of partial differential equations (PDEs)
- **PDEs are used to mathematically describe physical process**
- **Examples:**
  - **Navier-Stokes equations:**
    motion of fluids
  - **Continuity equations:**
    behavior of charge carriers in semiconductors
  - **Euler's equations (+Euler-Cauchy stress):**
    stress analysis in structural mechanics
  - **Etc.**







Meshes are Everywhere



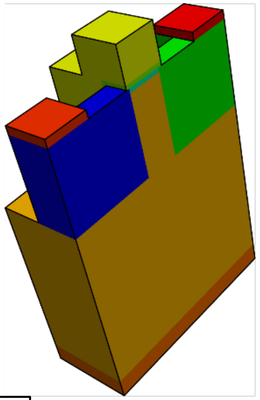
Source: Hang Si, WIAS, Berlin

Why is it Needed?

- PDEs are applied to a simulation domain including boundary conditions
- Boundary conditions represent influences from outside the simulation domain
- Analytic solutions are rarely possible
- Numerical approaches have to be used to calculate approximate solutions, most importantly:
  - Finite different method (FDM)
  - Finite volume method (FVM)
  - Finite element method (FEM)
- These methods require a discretized simulation domain: A <u>mesh</u>

Geometry

- **A simulation domain has to be specified by a geometry**
- **A geometry is not required to be <u>connected:</u>**
  But, there is always a finite partition of a geometry which consists of connected sets
- **Partitioning of simulation domains into <u>regions</u>, e.g., materials → multi-region geometry**



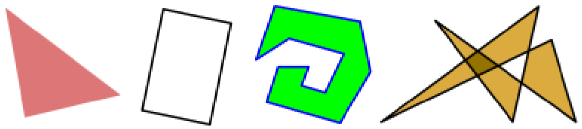
Definition (Geometry). Let $\mathcal{G} \subseteq \mathbb{R}^n$. $\mathcal{G}$ is called a geometry, if there are sets $G_1, \ldots, G_m \in \mathcal{L}^n$ (where $\mathcal{L}^n$ denotes the geometry space) which are connected and the geometry can be represented as a union of these sets: $\mathcal{G} = \bigcup_{i=1}^{m} G_i$.

Formal definitions not required for the exam!
But, informal description and understanding is.

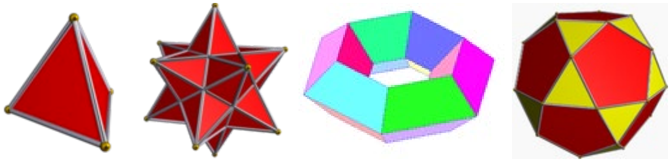
- **n-dimensional geometry: if $\mathcal{G} \in \mathcal{L}^n$ and if $\mathrm{DIM}(\mathcal{G}) = n$**
- **Representations:**
  - **Implicit representations**
  - **Boundary Representation and Piecewise Linear Complexes**
  - **Constructive Solid Geometries**

Polygon, Polyhedron, Polytope

- **Polygon:**
  Plane figure (2D) described by finite number of connected, straight line segments



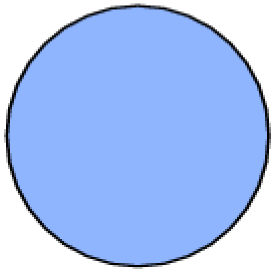
- **Polyhedron (sometimes *plural: Polyhedra*):**
  3D solid bounded by flat polygonal surfaces (analogous to a polygon in 2D)



- **Polytope:**
  Any geometric object with flat sides. Generalization of polygon/polyhedron in any number of dimensions:
  Polyhedron is a 3D polytope
  Polygon is a 2D polytope

Implicit Representation

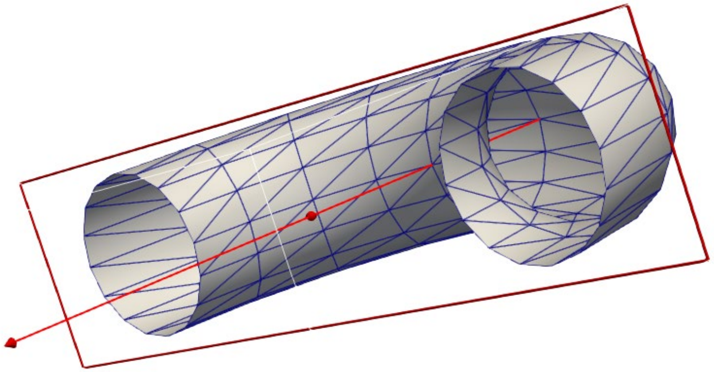
- Closed n-ball with radius r and center $\vec{0}$
- Represented using the function $F(\vec{x}) = \|\vec{x}\|_2 - r$

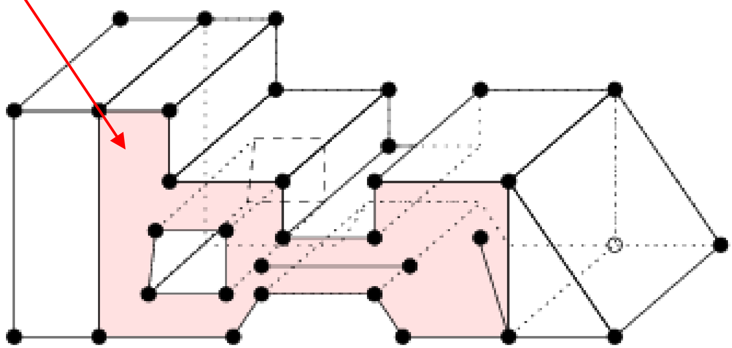


- Robust with respect to modeling
- However, finding a function in closed form is very challenging for general objects: Rarely used in engineering

Boundary Representation and Piecewise Linear Complexes

- Geometry $\mathcal{G} \subseteq \mathbb{R}^n$ with $\mathrm{DIM}(\mathcal{G}) = n$ can be represented by its boundary
- In 3D a boundary is typically described using piecewise linear functions:
  - Triangular hull (very common in computer graphics)
  - 3D piecewise linear complex (PLC) defined by 2D planar straight line graphs





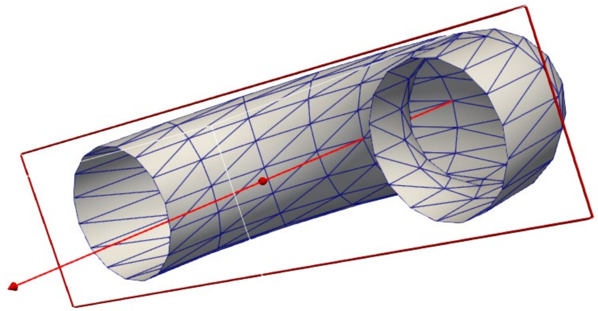
- Underlying space of PLC: linear geometry. Every polyhedron can be represented by a PLC.
- PLC $\mathcal{P}$ represented with boundary facets and additional hole and seed points: support multiple regions/holes.

Triangular Hull

https://www.paraview.org/

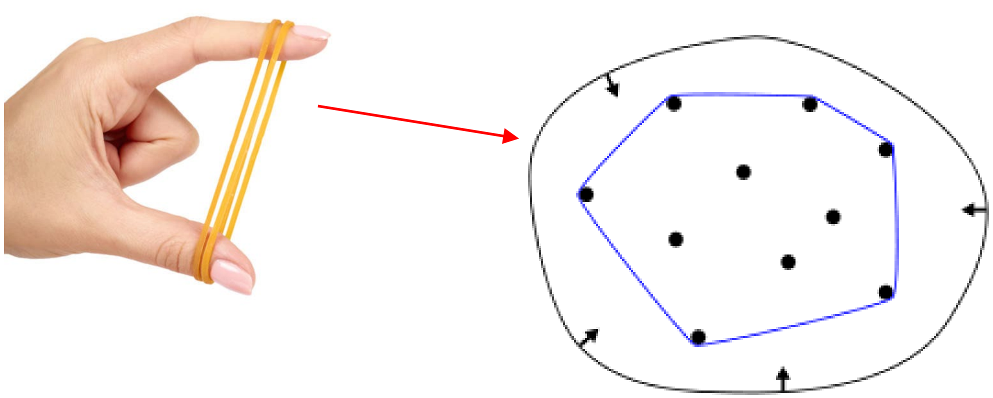
Guide:

1. Source → Superquadratic → Apply
2. Filter → Clip → Apply
3. Use visualization option "Surface With Edges"



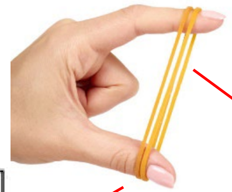
Convex Hull

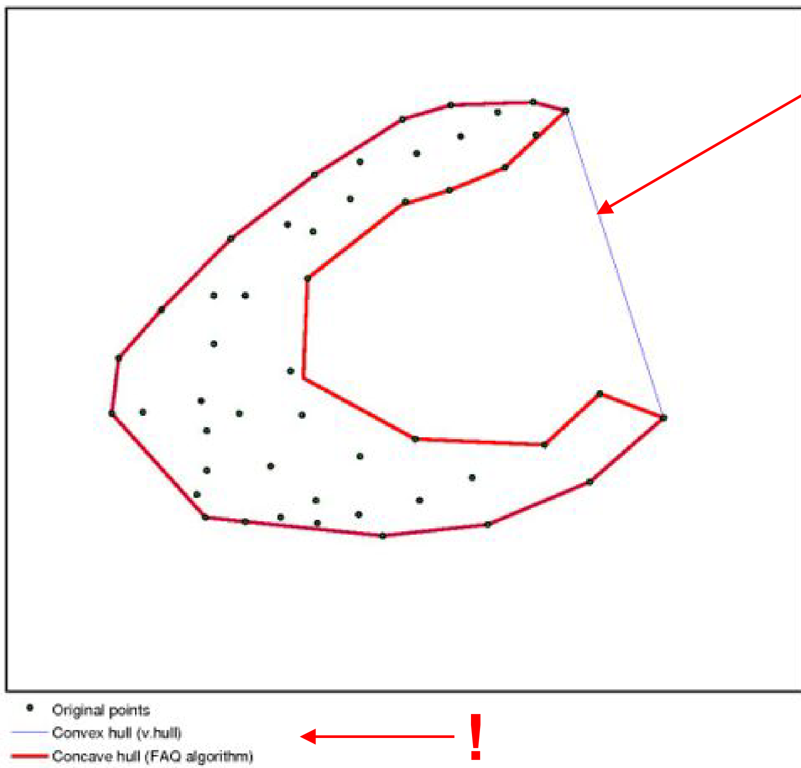
- The convex hull is the smallest enclosing convex polytope.
- X set of points in a Euclidian space
- X is defined to be convex if it contains the line segments connecting each pair of its points.
- If X is a bounded subset of the space, the convex hull may be visualized as the shape enclosed by a *rubber band* stretched around X.

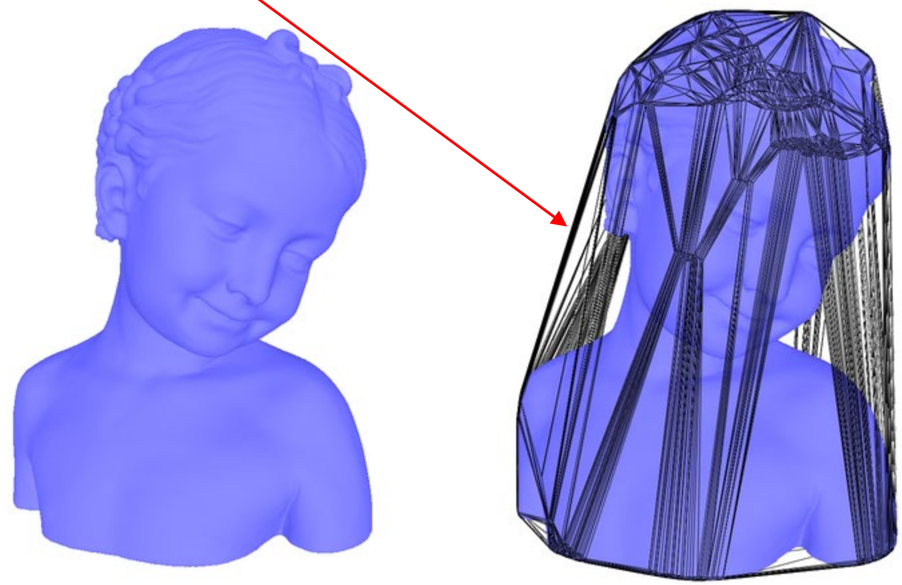


Convex vs Concave Hull

Concave hull or more general: *alpha shapes*
https://doc.cgal.org/latest/Alpha_shapes_3/





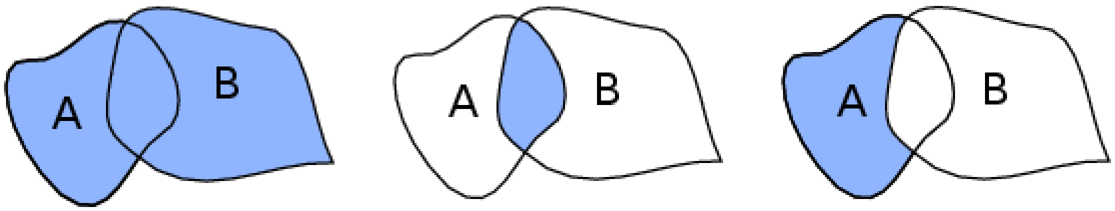


Source: GRASS-Wiki

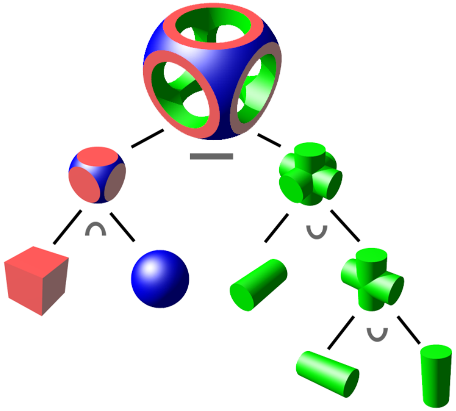
Source: CGAL

Constructive Solid Geometries

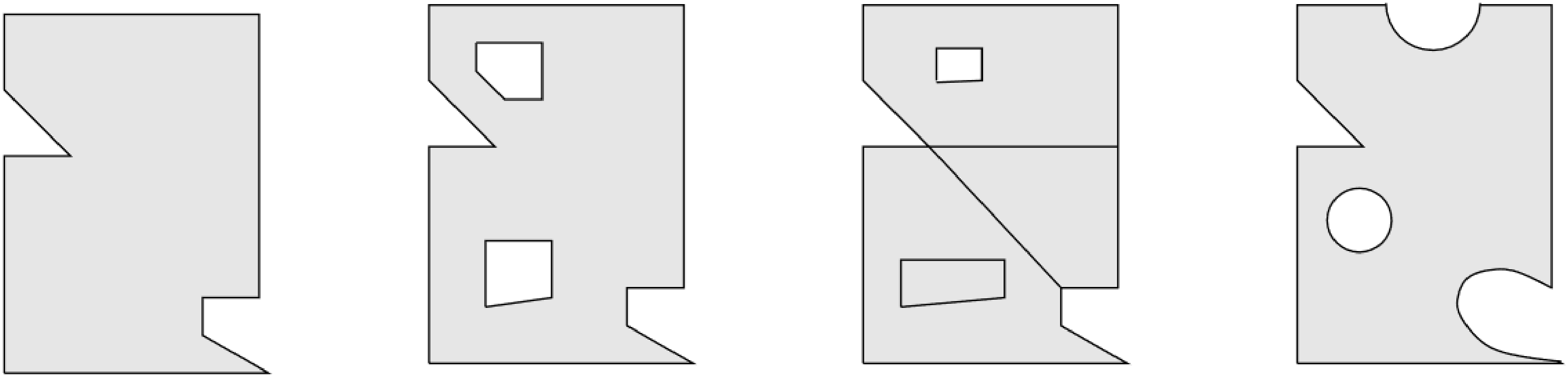
- Constructive solid geometry (CSG)
- Basic shapes, like cubes or spheres, are used together with set operations to represent a geometry
- Commonly supported set operations are set intersection, set union, and set difference



- Basic shapes and set operations are used to form a hierarchical CSG tree
- CSG allows for rapid-prototyping



Types of 2D Geometries

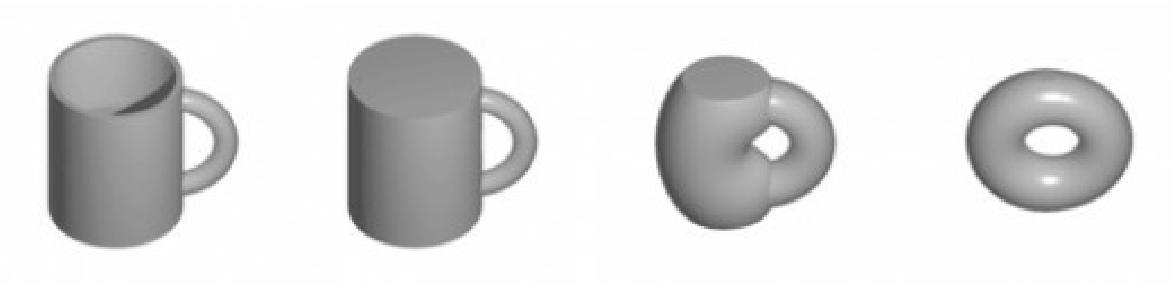


- Simple polygon
- Polygon with holes
- Multiple domain
- Curved domain

Topology

- **Informally: Concerned with properties of geometric object that are preserved under continuous deformations (e.g. stretching, twisting, cumpling, bending) but not tearing or gluing.**

"A topologist cannot distinguish a coffee mug from a doughnut."



Topology vs Geometry

Geometry

- **Where elements are placed in space**

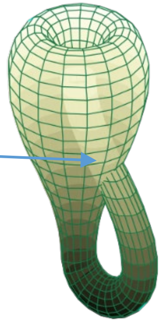
Topology

- **How are the elements connected and oriented**

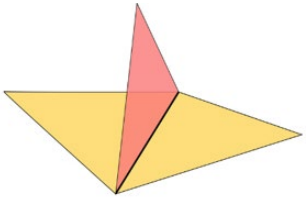
Manifold

- **A manifold is a topological space that locally resembles Euclidian space**
- **Lines and circles are 1D manifolds**
- **2D manifolds are also called surfaces:**
  $\mathbb{R}^3$: **Sphere, torus,**
  Moebius strip (2D manifold with boundary, can be embedded in Euclidian space $\mathbb{R}^3$)

  $\mathbb{R}^4$: **Klein bottle** (2D manifold without boundary, can be embedded in Euclidian space $\mathbb{R}^4$ but not in $\mathbb{R}^3$! Important: The *intersection* is not really there!)

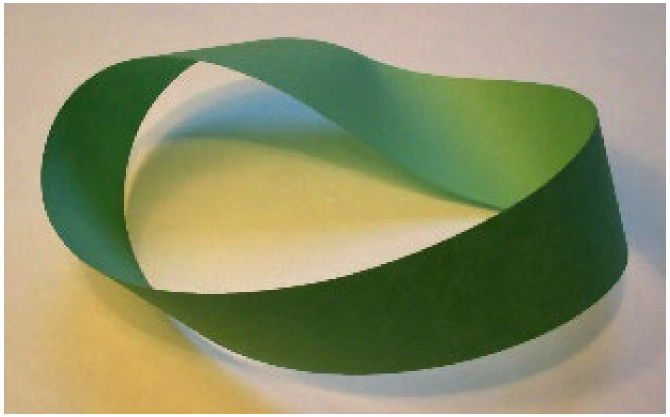


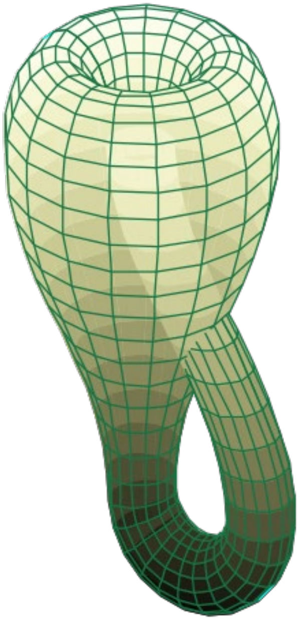
- **Non-manifolds**
  - **Figure eight** (because no neighborhood of crossing point – an intersection – resembles Euclidean 1-space)
  - **Two intersecting planes** (because neighborhood along intersection doesn't resemble a Euclidian 2-space)



Orientability

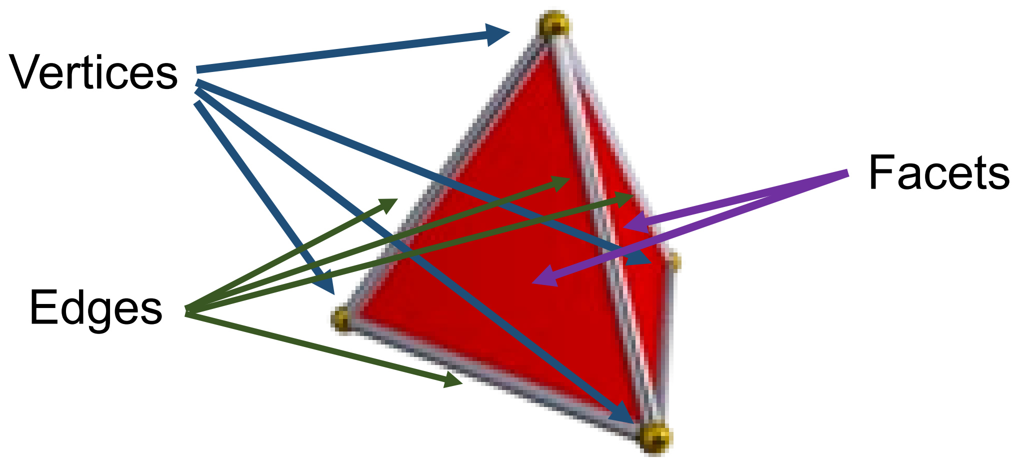
- A surface is orientable if it is possible to make a consistent choice for the normal vector (aka the surface has “two sides”)
- Quick poll:
  Is the Moebius strip orientable?
  Is the Klein bottle orientable?
- Moebius strip, Klein bottle, and non-manifold surfaces are not orientable





Vertex, Edge, Face, and Facet

- **Vertex is a corner point of a polytope formed by the intersection of edges or faces of the object**
- **Edge is a line segment joining two vertices in a polytope.**
- **Fac<u>et</u> of a polytope of dimension n is a face that has dimension $n-1$. E.g.:Tetrahedron → triangles (2-faces)**
- **Face of dimension k is called a k-face.**
  Cube: vertices (0-faces), edges (1-faces), facets (2-faces), cube itself (3-face)

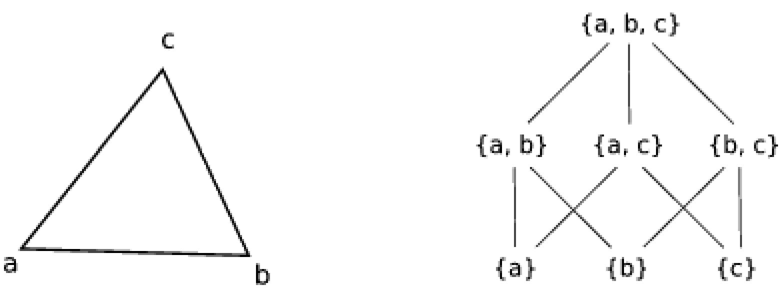


Simplex and Polytope

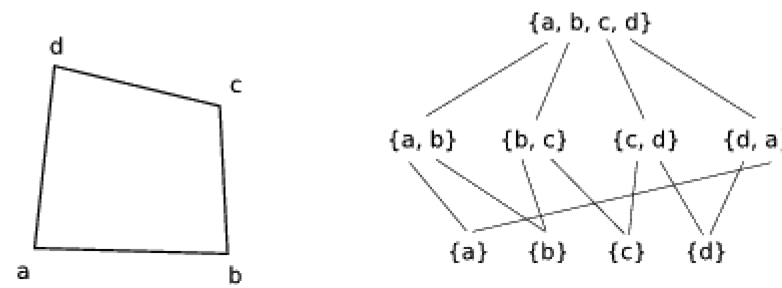
- **A simplex is the generalization of the notion of a triangle to arbitrary dimensions**
- **A simplex is the *<u>simplest</u>* *polytope* in a given dimension**
- **A k-simplex is a k-dimensional polytope with the convex hull of its $k+1$ vertices.**
  - **-1-simplex: $\emptyset$**
  - **0-simplex: point**
  - **1-simplex: line**
  - **2-simplex: triangle**
  - **3-simplex: tetrahedron**

Generalization: Cell

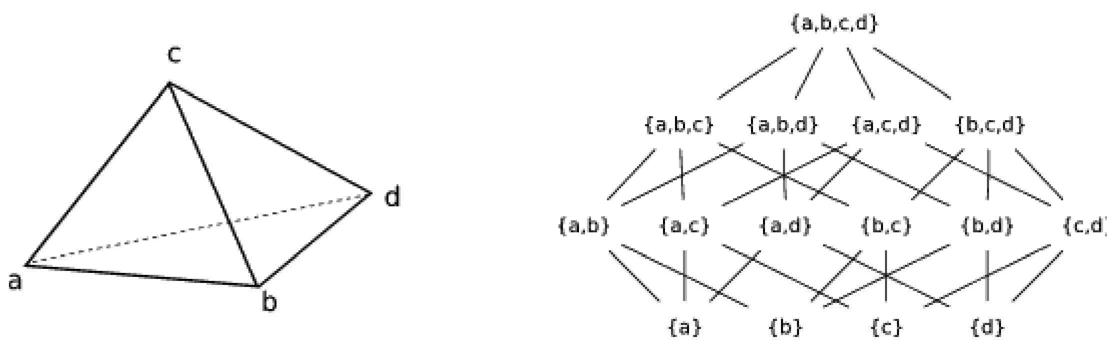
- **Cell is a convex polytope**
- **A p-dimensional p-cell can describe, e.g., a simplex or a cuboid cell**
- **Some cell topologies (*Hasse* diagrams; omitting $\emptyset$):**



Simplex 2-cell



Cuboid 2-cell



Simplex 3-cell

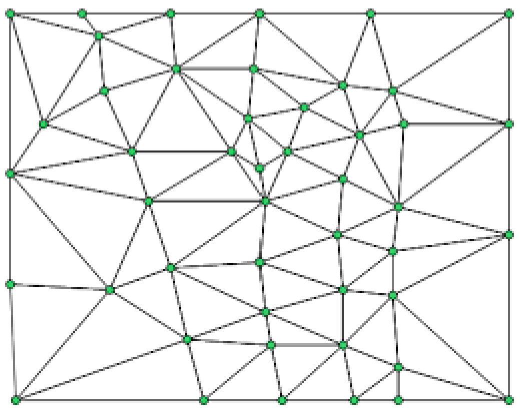
Cell (CW) Complex

Prelude to mesh definitions

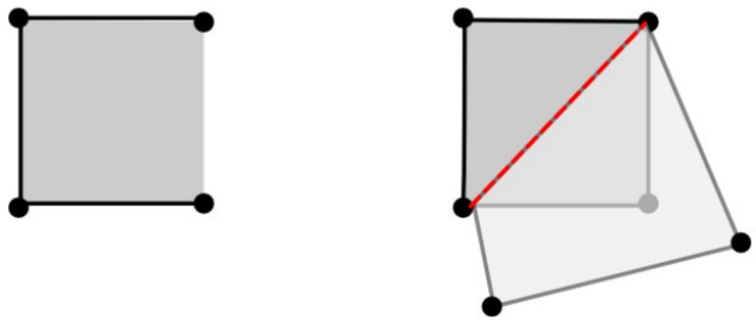
A collection of cells is a *cell complex iff*

- **Every face of a cell belongs to the complex**
- **For every cell pair C and C’, their intersection is either empty or is a common face of both**

Cell complex* is a *simplicial complex* when all cells are *simplices



Cell Complex



Violations

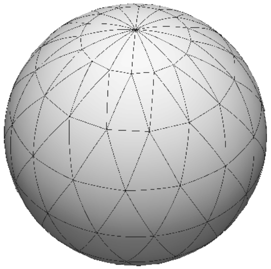
Maximal Cell Complex

- **Cell order (=dimension)** (e.g. how many dimensions do you need to draw a square? A: 2 → 2-cell)
- **Cell complex is a k-complex if the maximum of the order of its cells is k** (e.g. a set of 2-cells represents a 2-complex)
- **Cell is maximal if it is not a face of another cell** (e.g. a triangle of 3D tetrahedral volume mesh is not a maximal cell; here only the tetrahedrons would be maximal cells)
- **K-complex is maximal *iff* all maximal cells have order k**
- **Triangle mesh → maximal 2-simplicial complex**
- **Tetrahedral mesh → maximal 3-simplicial complex**

Adjacency and Incidency

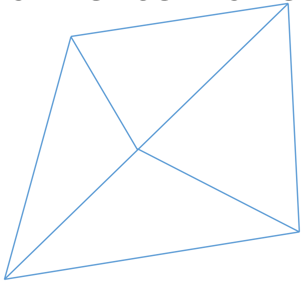
Incidency

- Two cells are incident if one of them is a proper face of the other
  - In a closed manifold triangular mesh each edge has exactly two incident triangles and each triangle has three incident edges



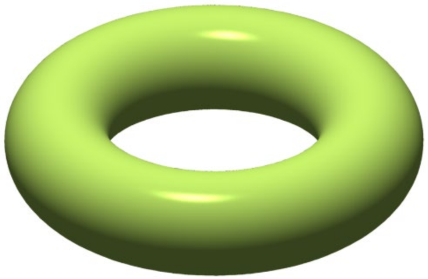
Adjacency

- Two k-cells are m-adjacent (k>m) if there exists a m-cell that is a proper face of both k-cells. Examples:
  - Two triangles sharing an edge are 1-adjacent
  - Two triangles sharing a vertex are 0-adjacent



Genus

- **The Genus (g) of a surface is the largest number of nonintersecting simple closed curves that can be drawn on the surface without separating it.**
  (informally: the number of “holes”)



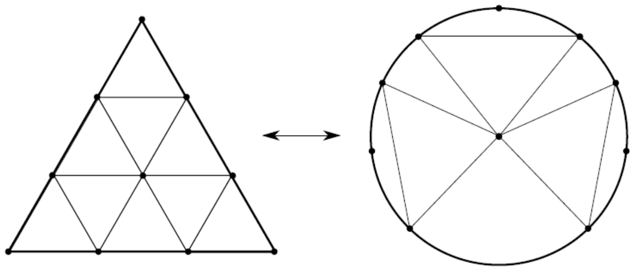
$g = 1$

$g = 2$

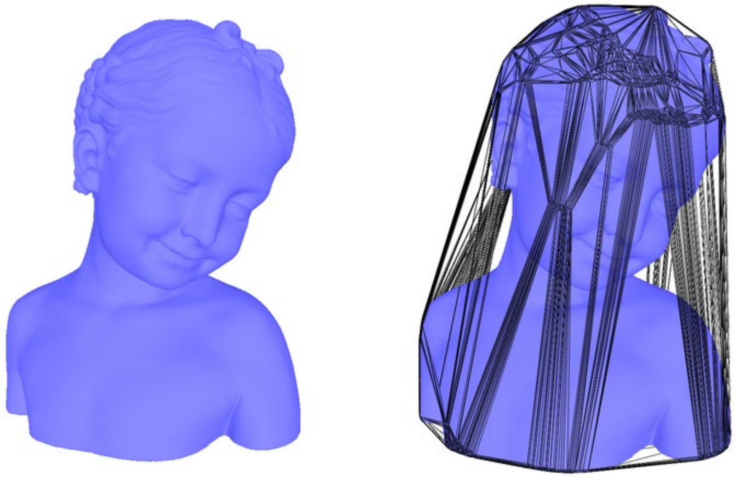
Quick Poll!

Triangulation

- **Informal: a tessellation of an object**
- **Triangulations may be topological equivalent**



- ***Sometimes* a triangulation uses the convex hull: problematic for concave objects**

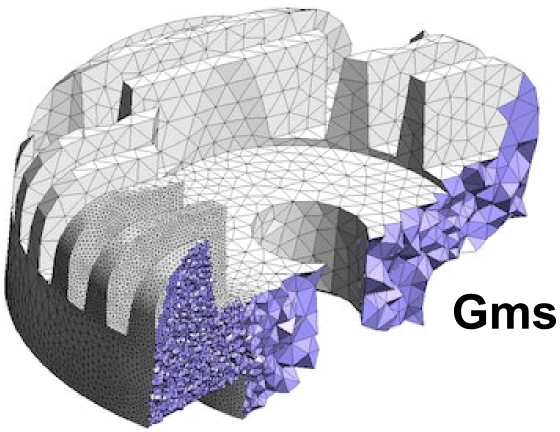


Triangulation vs Mesh: A Source of Confusion

Mesh is

- **… sometimes considered an "engineering term"**
- **… sometimes referred to as "grids"**
- **… a *conforming* triangulation: $\boxed{\text{preservation of boundary}}$ (see previous discussion on convex vs concave)**
- **… discrete representation Ω_h of a continuous domain Ω**
- **... the union of non-overlapping closed subdomains Ω_h^k created by partitioning the domain into K smaller elements such that:**

$$\Omega \cong \Omega_h = \bigcup_{k=1}^{K} \Omega_h^k$$



Gmsh

Meshing: Non-Boundary Preservation

https://www.paraview.org/

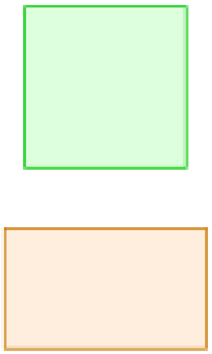
Guide:

1. Source → superquadric → Apply
2. Filter → Delaunay3D → Apply
   Note: convex triangulation of a concave object → hole closed.
3. Use visualization option “Surface With Edges”

2D Meshing Elements

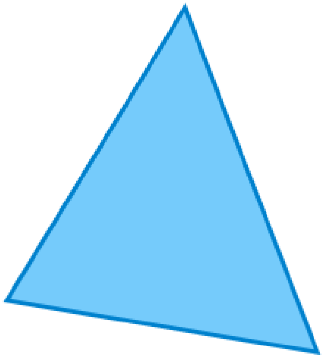
Quadrilateral (cuboid 2-cell)

- **A polygon with four edges**
- **Square is a regular quadrilateral with equally long edges**



Triangle (simplex 2-cell)

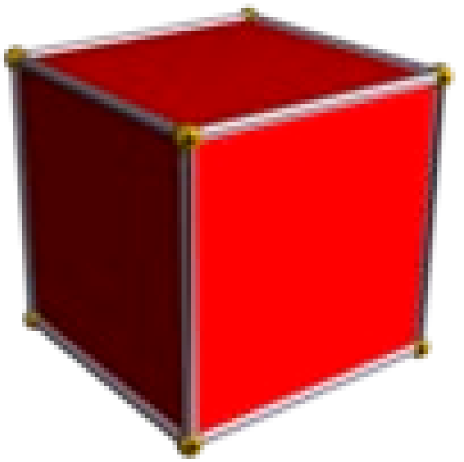
- **A polygon with three edges**



3D Meshing Elements

Hexahedron (cuboid 3-cell)

- A polyhedron with six quadrilateral faces
- Cube is a regular hexahedron with square faces



Tetrahedron (simplex 3-cell)

- A polyhedron with four triangular faces, six straight edges, and four vertex corners

3D Mesh

https://www.paraview.org/

Guide:

1. **Source → Sphere → Apply**
2. **Filter → Delaunay3D → Apply**
3. **Filter → Clip → Apply**
4. **Use visualization option "Surface With Edges"**

Types of Meshes

Structured mesh

- **All interior vertices are topologically alike**
- **Simple and easy data access**
- **Mesh elements: quadrilaterals (2D), hexahedra (3D)**

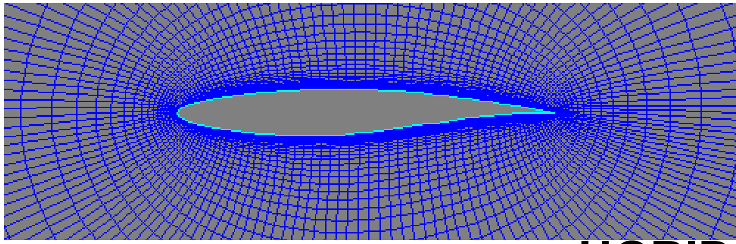
Unstructured mesh

- **Vertices may have arbitrarily varying local neighborhoods**
- **Favors mesh adaptation and support for complicated domains**
- **Mesh elements: triangles (2D), tetrahedra (3D)**

Hybrid mesh

- **Number of structured meshes combined in an overall unstructured pattern**

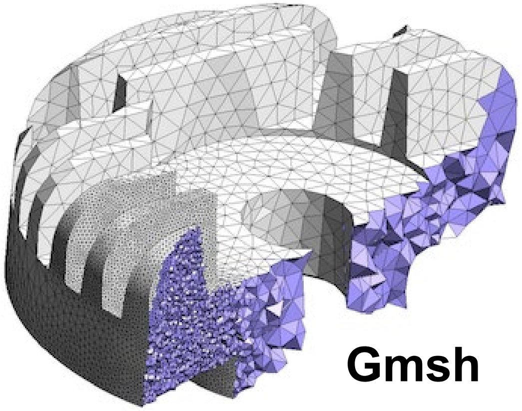
Structured Mesh


UGRID

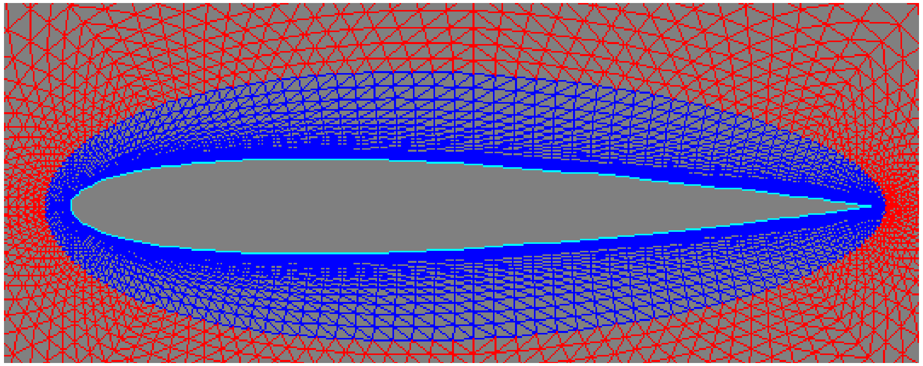
- **Offers simplicity and efficiency**
- **Requires significantly less memory than unstructured meshes: Array storage implicitly defines neighbor connectivity**
- **Efficient because neighbor access requires increments/decrements of array indices (and compilers can optimize these access patterns)**
- **However, problematic to represent complicated geometric domain**
- **Also, requires many more elements than unstructured meshes because elements cannot grade in size rapidly**

Unstructured Mesh

- Flexibility in fitting complicated domains
- Rapid grading from small to large elements
- *Relatively* easy refinement / de-refinement (aka coarsening)
- Unstructured mesh generation: Delaunay triangulation
- *Simplex complex*



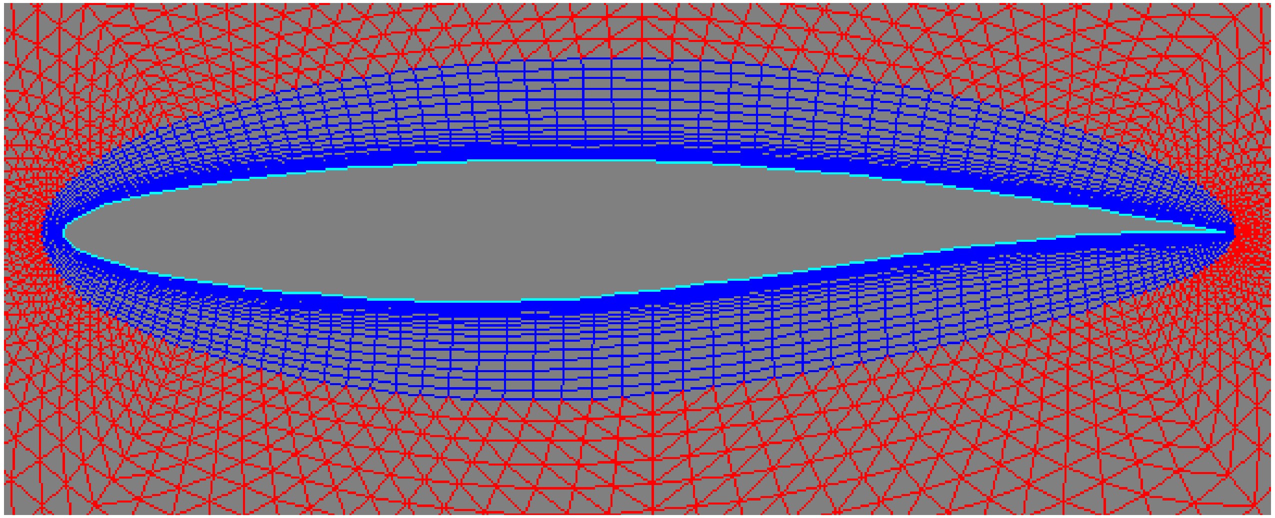
Gmsh



UGRID

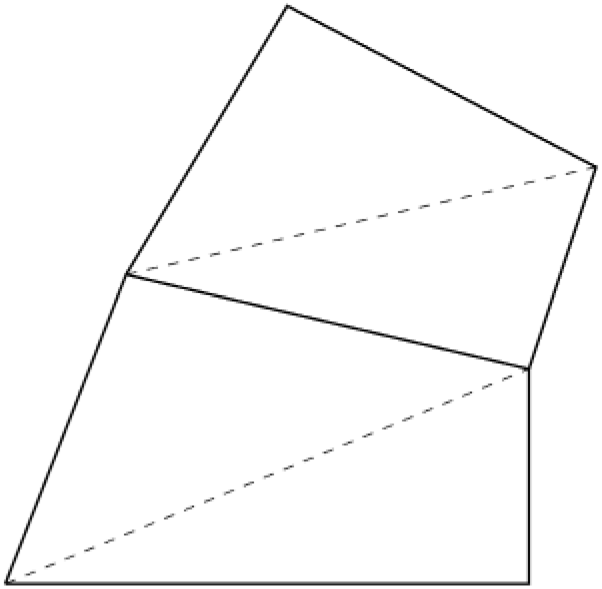
Hybrid Meshes

- **Combines advantages of both approaches**
- **Not straightforward to generate**

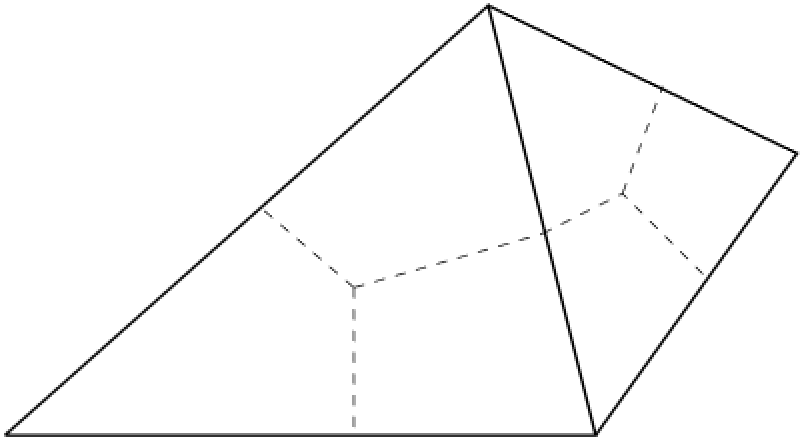


UGRID

Triangles ⇔ Quadrilaterals



Triangulating quadrilaterals



Subdividing triangles to form quadrilaterals

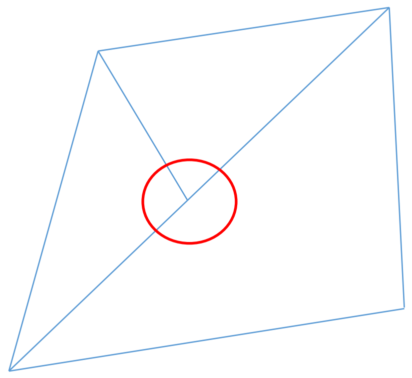
Mesh Conformity

Conformal mesh

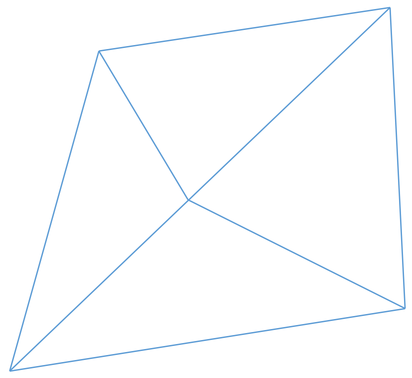
- Vertices, edges, faces of neighboring elements are perfectly matched

Hanging vertices

- Vertices which are not perfectly matched with a neighboring vertices

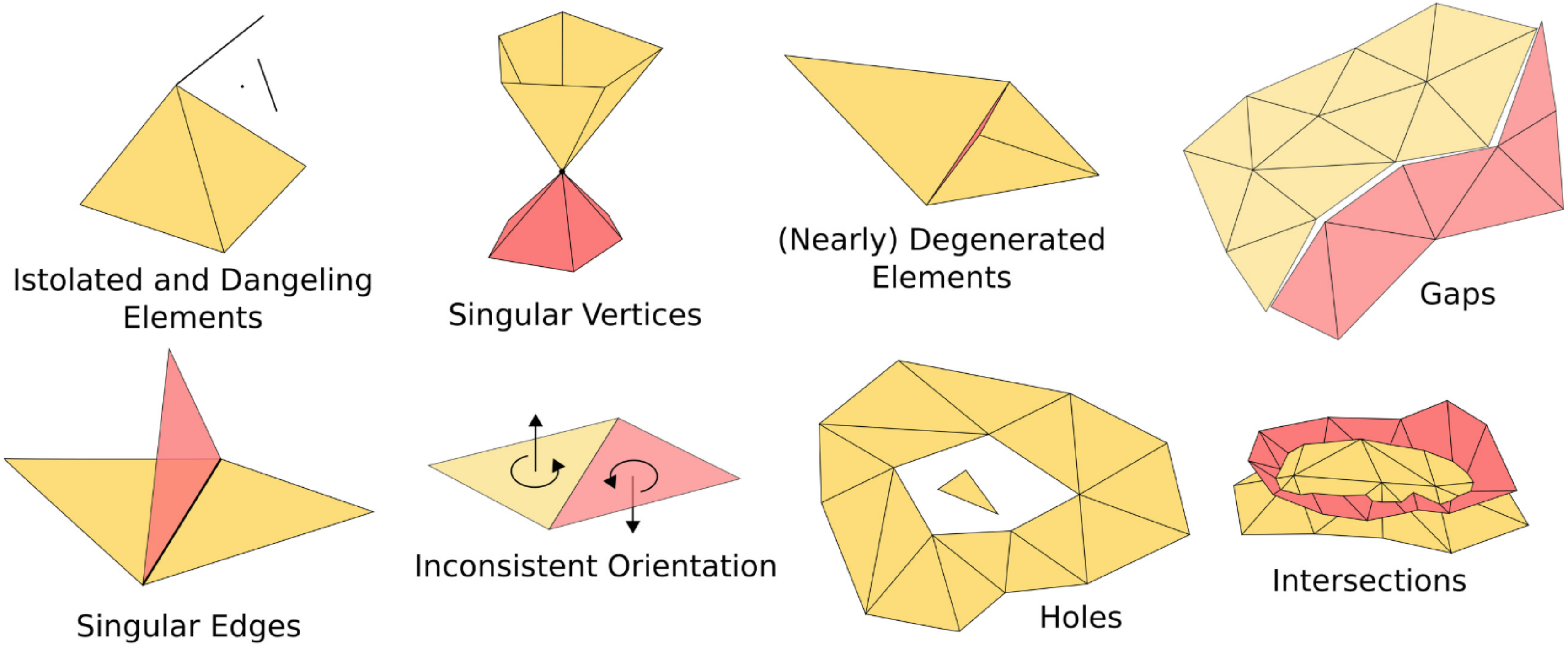


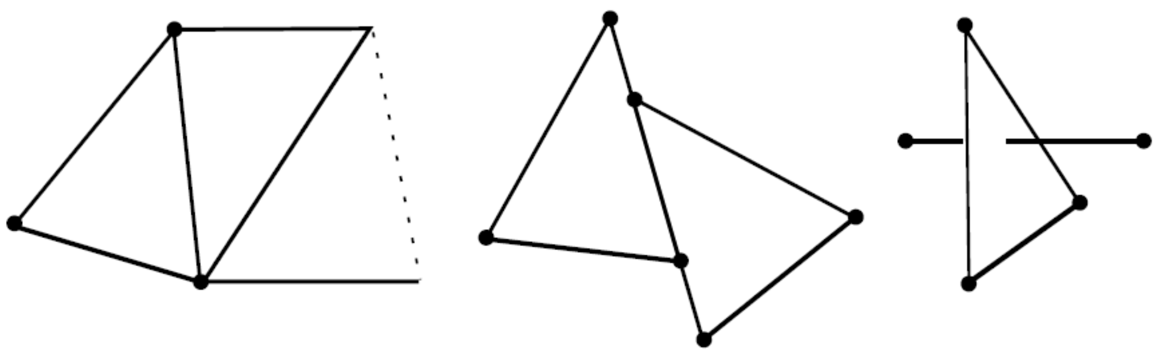
Non-conforming mesh:
Hanging vertex



Conforming mesh

Conformity Examples: (3D Hull) Mesh Errors

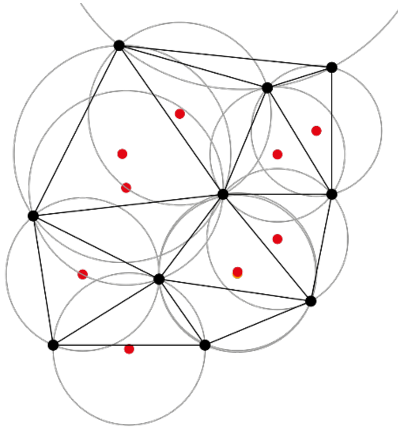




Delaunay Triangulation (incl. Meshing)

- **Delaunay triangulations maximize the minimum angle of all the angles of the triangles**
- **Tends to avoid sliver triangles (triangle with one or two extremely acute angles)**

Definition (Delaunay Triangulation). A Delaunay triangulation of a vertex set is a triangulation of the vertex set with the property that no vertex in the vertex set falls in the interior of the circumcircle of any triangle in the triangulation.



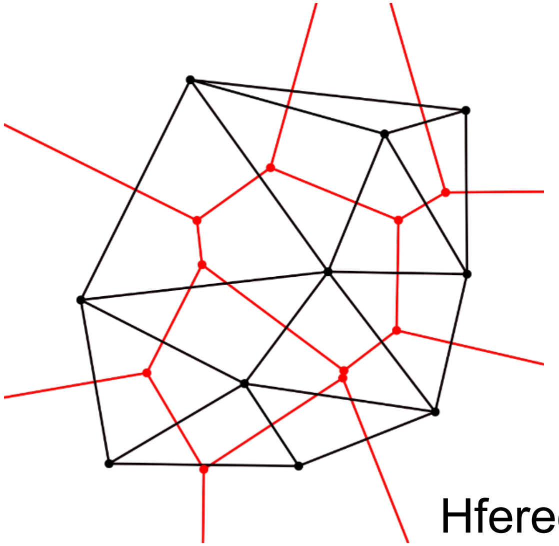
Matthias Kopsch, Wikipedia

Delaunay Triangulation and Voronoi Diagram

- **A Delaunay triangulation of a vertex set corresponds to the dual graph of the Voronoi diagram.**

Voronoi diagram

- **The partitioning of a plane with n points into convex polygons such that each polygon contains exactly one generating point and every point in a given polygon is closer to its generating point than to any other.**
- **Important for finite volume method (FVM)!**



Hferee, Wikipedia

Mesh Quality

- Mesh generation is not unique:
  - Different algorithms yield different results
  - Different parameters yield different results
- Numerical simulation methods are very sensitive to mesh quality
- One very bad mesh element can be a problem for the solver! (much more problematic than several "average" elements) → always investigate the "worst" elements!

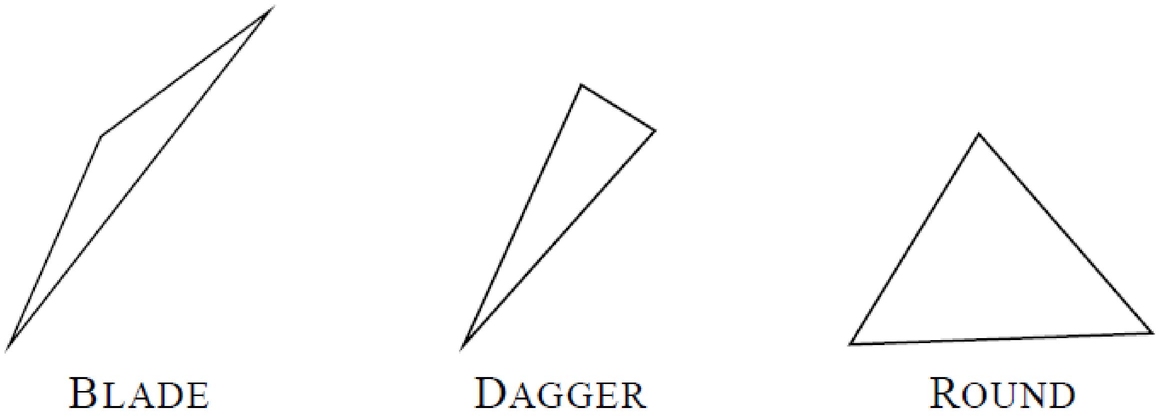
What is a "bad" mesh element?

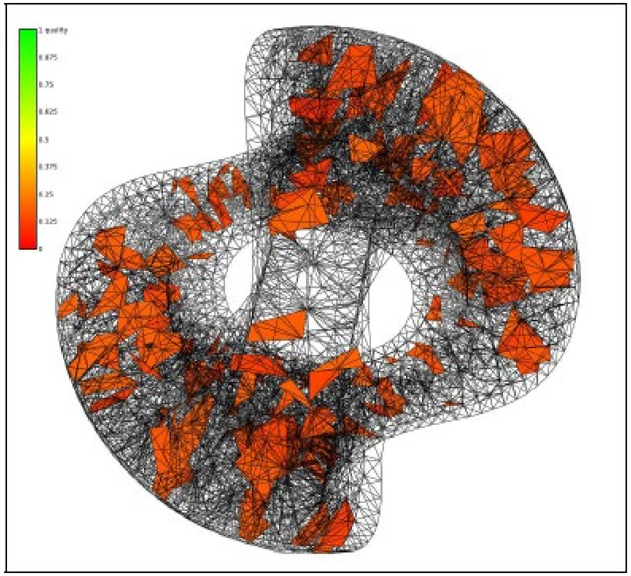
- large or small angles: introduce all kinds of errors in discretization and solution approaches

How to measure the quality of a mesh element?

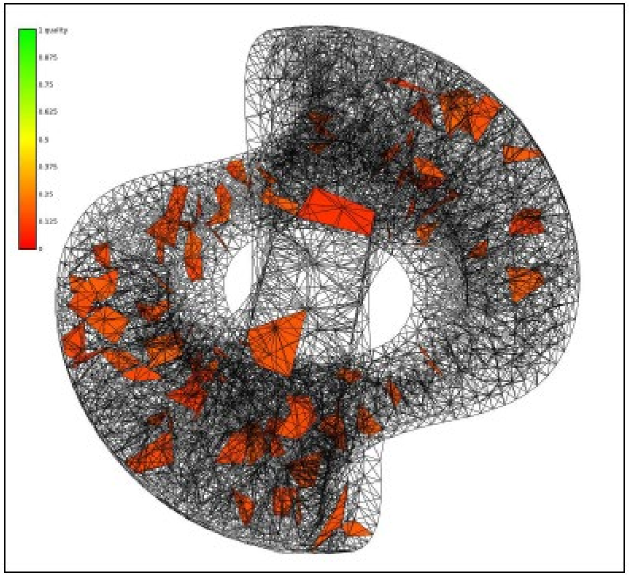
- Different metrics (ratios) are available
- E.g. volume/area to length ratio

Mesh Quality





Original



Refined

Mesh Quality: Why It Is Tough!

ParaView

https://www.paraview.org/

Task: Find the “one” bad triangle in the mesh!

Guide:

1. Open “NSSC_I_Lecture_10_bunny.ply” → Apply
2. Filter → Mesh Quality → Apply
3. Filter → Threshold → Apply
   Lower Threshold: 1000
   “Zoom on Data”
4. Turn on visibility of “NSSC_I_Lecture_10_bunny.ply”

Mesh Data Structures

- **How to store geometry and connectivity?**
- **Need to consider efficient algorithms on meshes!**
  E.g. all incident cells of a vertex
- **Face set (STL):**
  triangle list, positions only, no connectivity
- **Shared vertex (OBJ, OFF):**
  position list of vertices, list of faces using vertex indices
- **And others:**
  (half)edge/face-based connectivity

Vertices
x_1 y_1 z_1
...
x_V y_V z_V

Triangles
v_{11} v_{12} v_{13}
...
...
...
...
v_{F1} v_{F2} v_{F3}

OBJ/OFF

Triangles		
x_{11} y_{11} z_{11}	x_{12} y_{12} z_{12}	x_{13} y_{13} z_{13}
x_{21} y_{21} z_{21}	x_{22} y_{22} z_{22}	x_{23} y_{23} z_{23}
...	...	...
x_{F1} y_{F1} z_{F1}	x_{F2} y_{F2} z_{F2}	x_{F3} y_{F3} z_{F3}

STL

Free Open Source 3D Mesh Generation Tools

- **NetGen (LGPL) – Prof. Schöberl, TU Wien**
  https://ngsolve.org/
  GUI + API C++, Python, Jupyter Notebook
  Windows, Linux, macOS
- **Gmsh (GPL)**
  http://gmsh.info/
  GUI + API C/C++, Python, Julia
  Windows, Linux, macOS
- **CGAL**
  https://www.cgal.org/
  Library, API C++, Python
  Windows, Linux, macOS
- **TetGen (AGPLv3)**
  http://wias-berlin.de/software/index.jsp?id=TetGen&lang=1
  CLI Application + API C++
  Windows, Linux, macOS

Outline

- Quiz Wrapup
- Mesh Generation
- **Next Quiz**

New Quiz

1. What is the Hasse diagram of a cuboid 3-cell?
2. Is the letter “A” a convex or a concave object?
3. What is the Genus of the surface of a coffee mug?
4. Do you know a platform independent build system?
5. What is gdb?