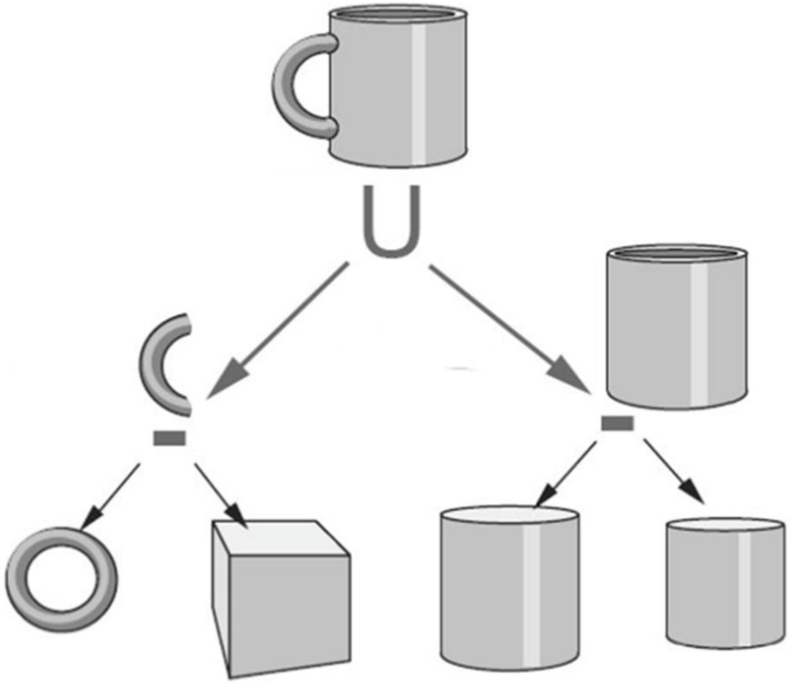
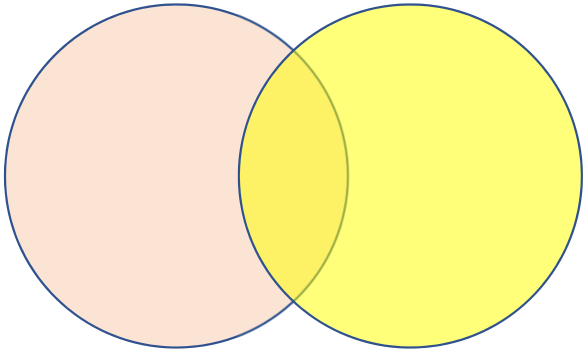
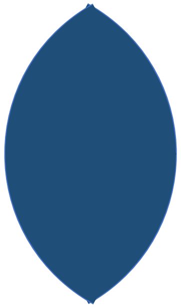
Constructive Solid Geometry

- Boolean binary operations on primitives
- How to do exactly? Easy if used in …
  - Ray Tracing
  - Voxels
- But how about meshes?
- How to compute the union, intersection and subtraction of two meshes?

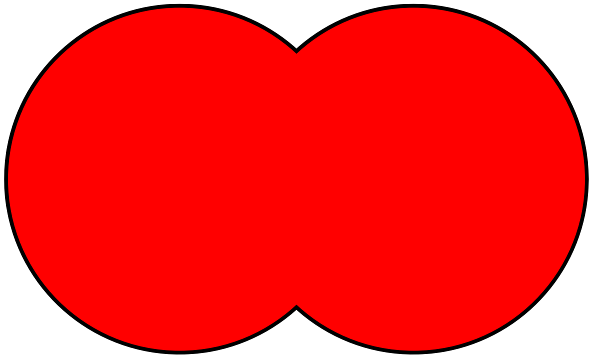




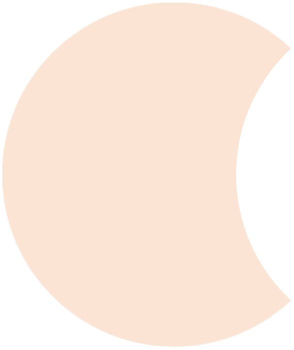
Set A and Set B



Intersection



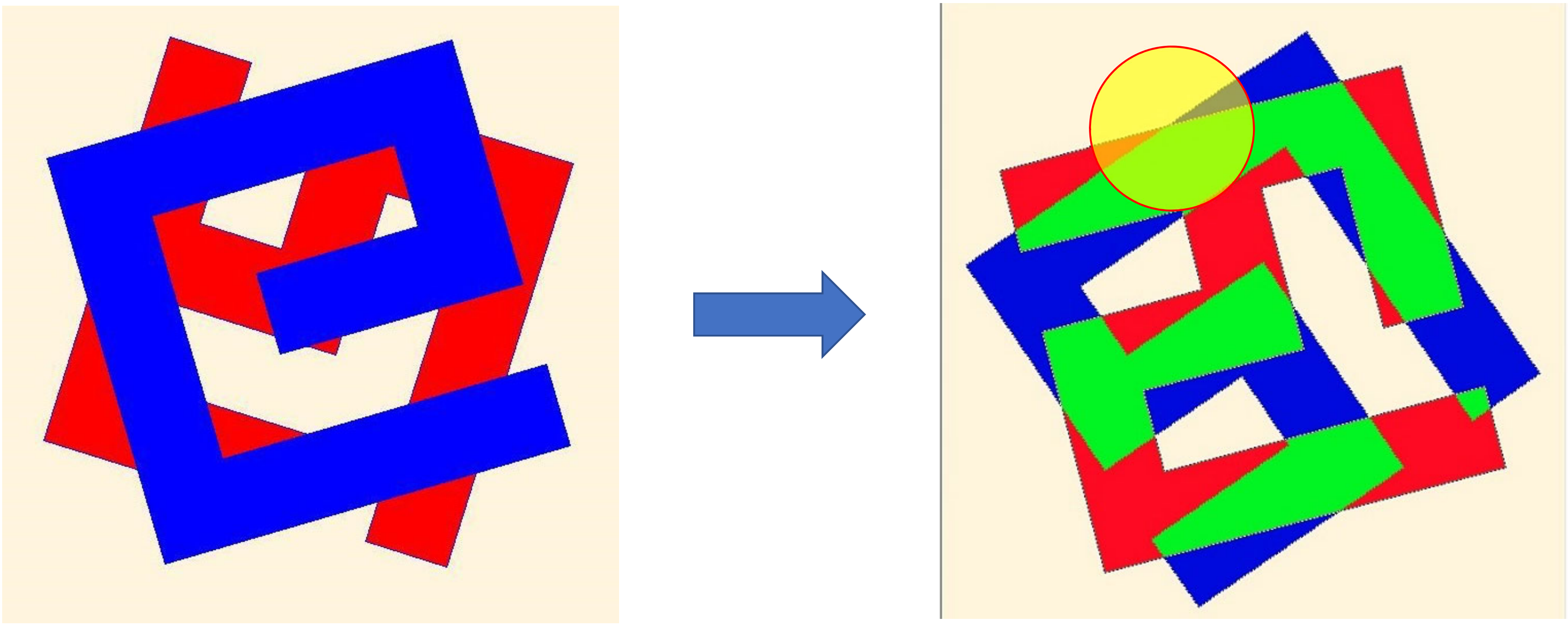
Union



A - B

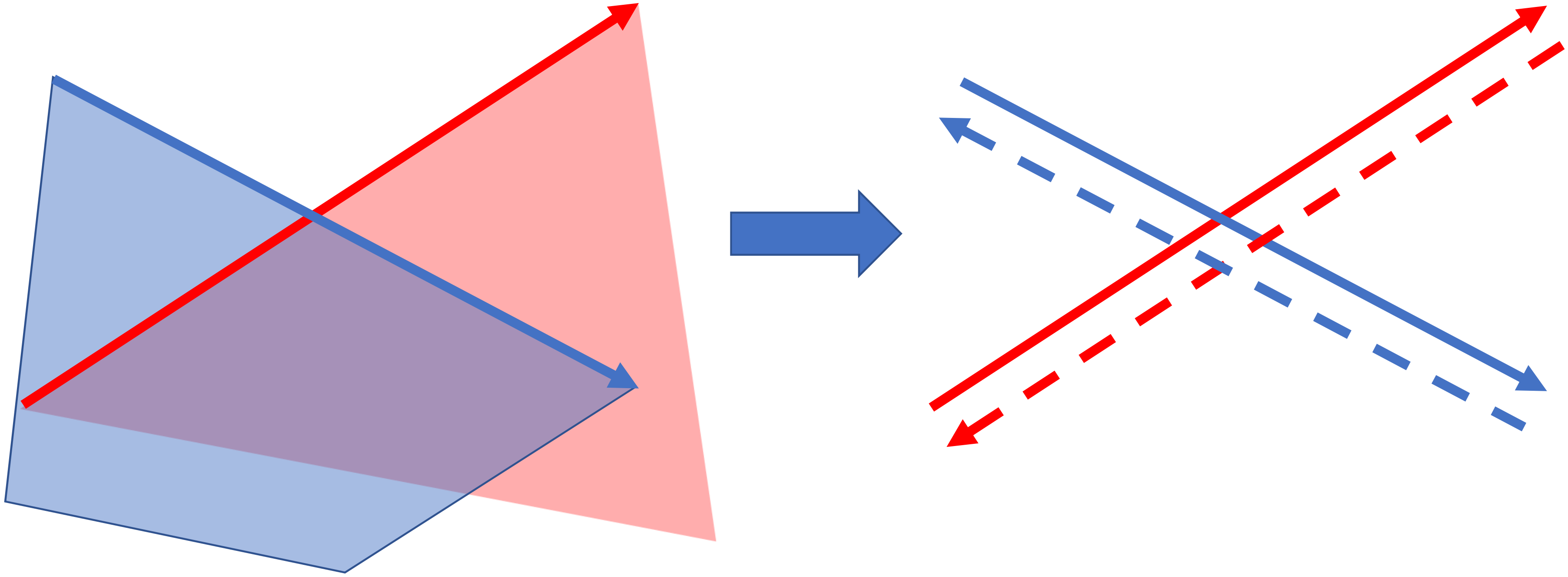
Weiler–Atherton Clipping Algorithm

- Let’s think about 2D first



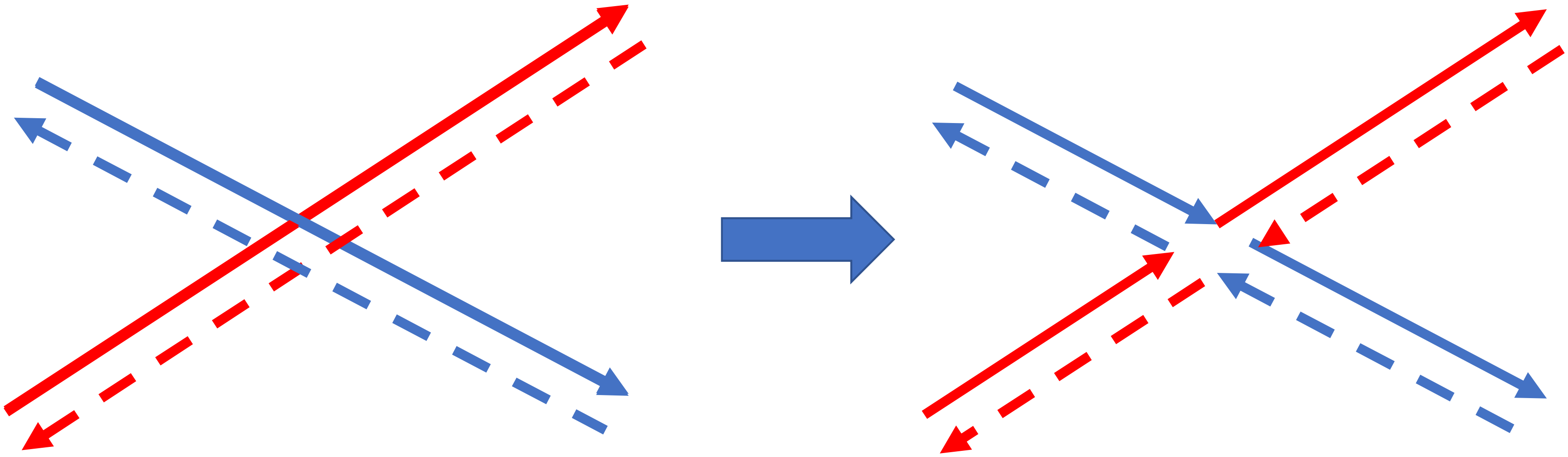
Weiler–Atherton clipping algorithm

- Every intersection
    - Solid = outside, dash = inside



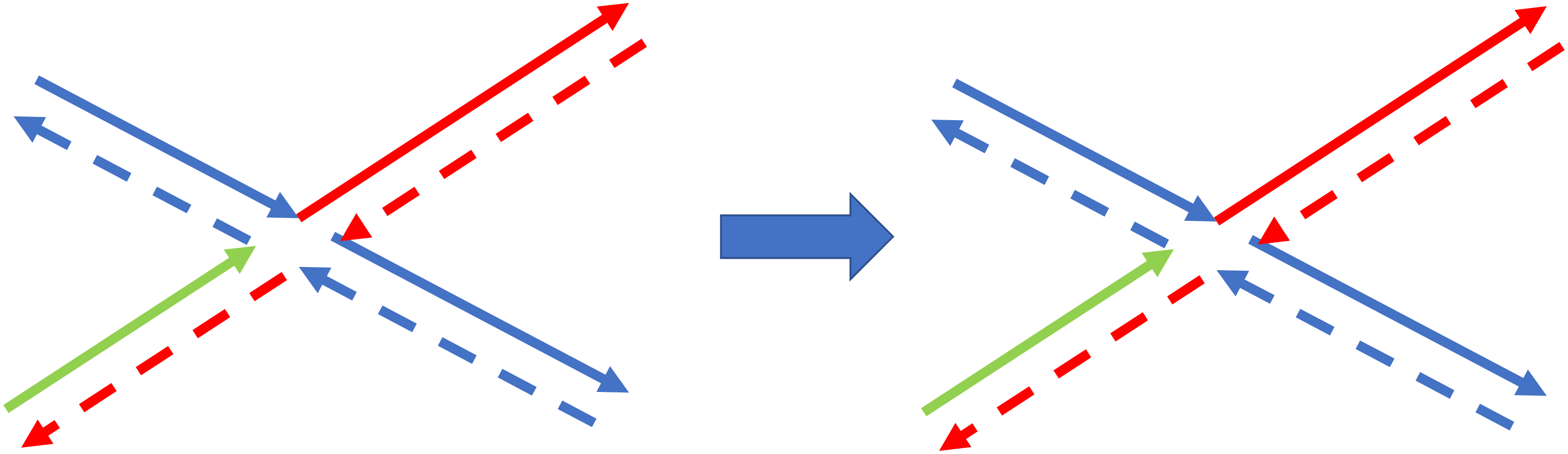
Weiler–Atherton clipping algorithm

- Every intersection
  - Solid = outside, dash = inside



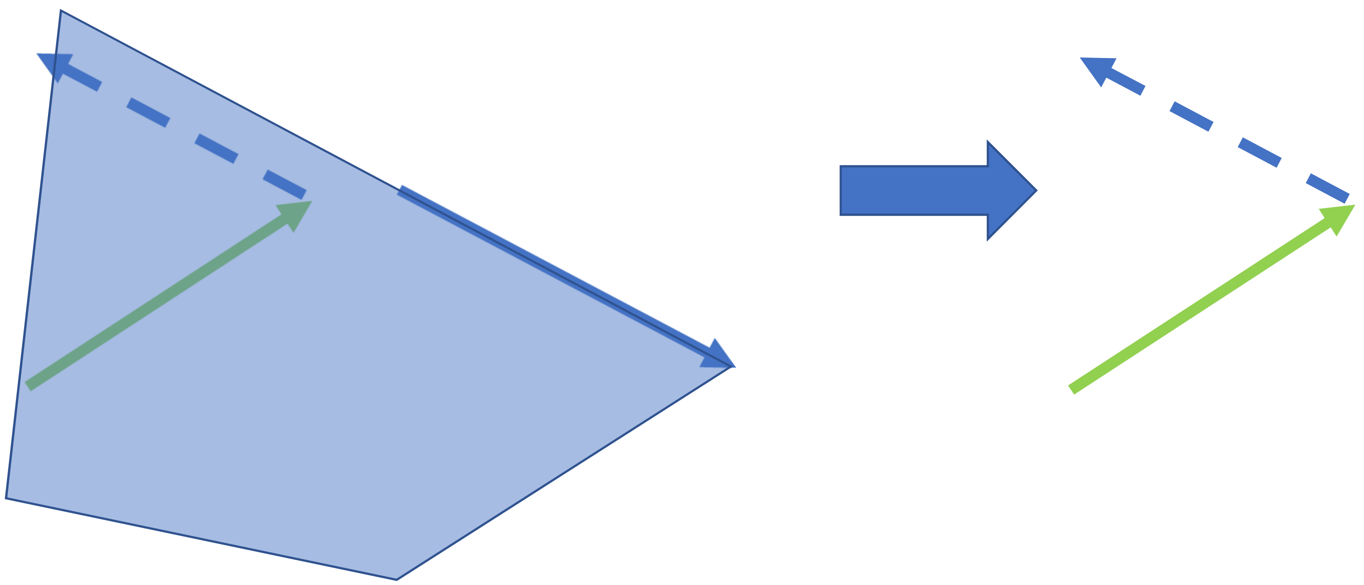
Weiler–Atherton clipping algorithm

- Every intersection
  - Solid = outside, dash = inside

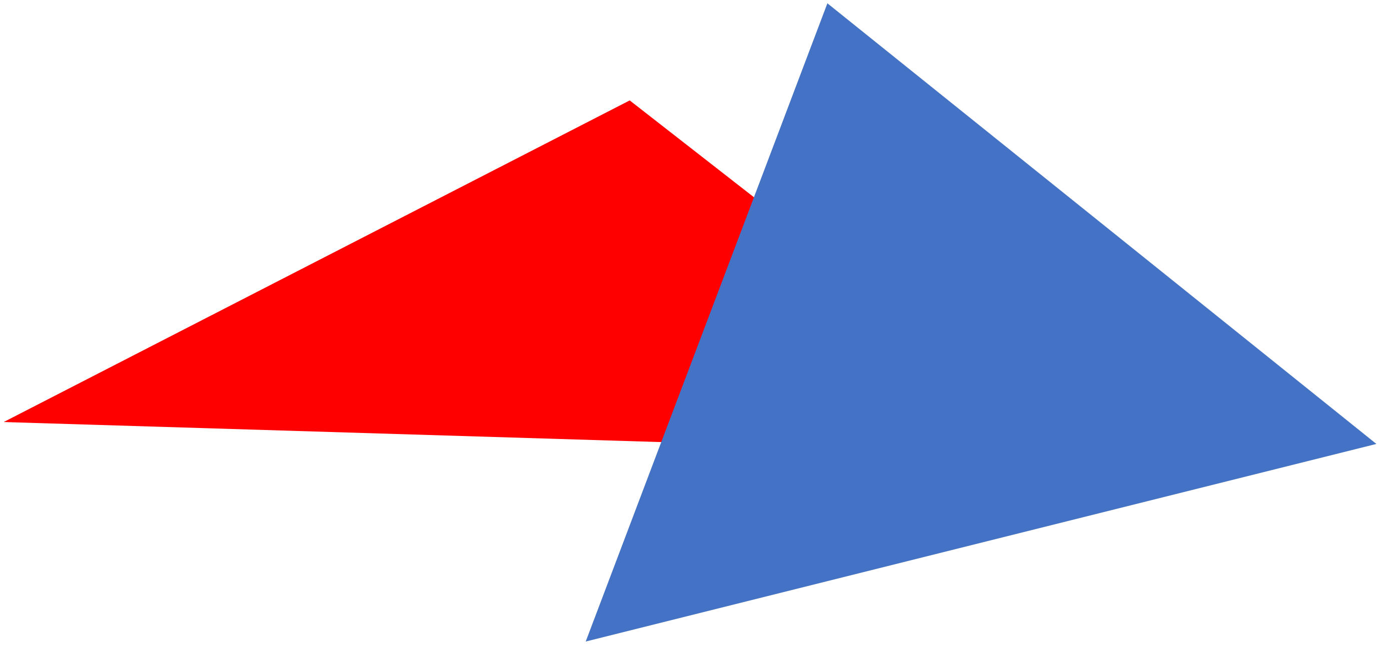


Weiler–Atherton clipping algorithm

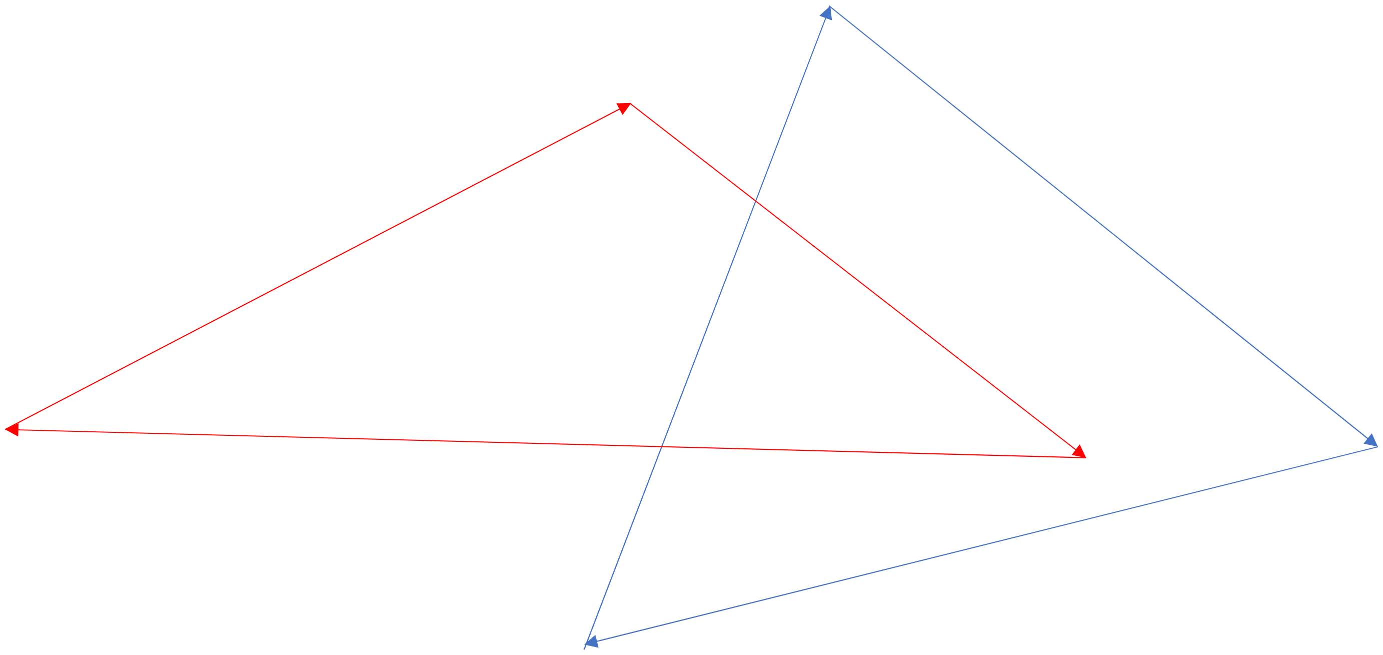
- Every intersection
  - Solid = outside, dash = inside



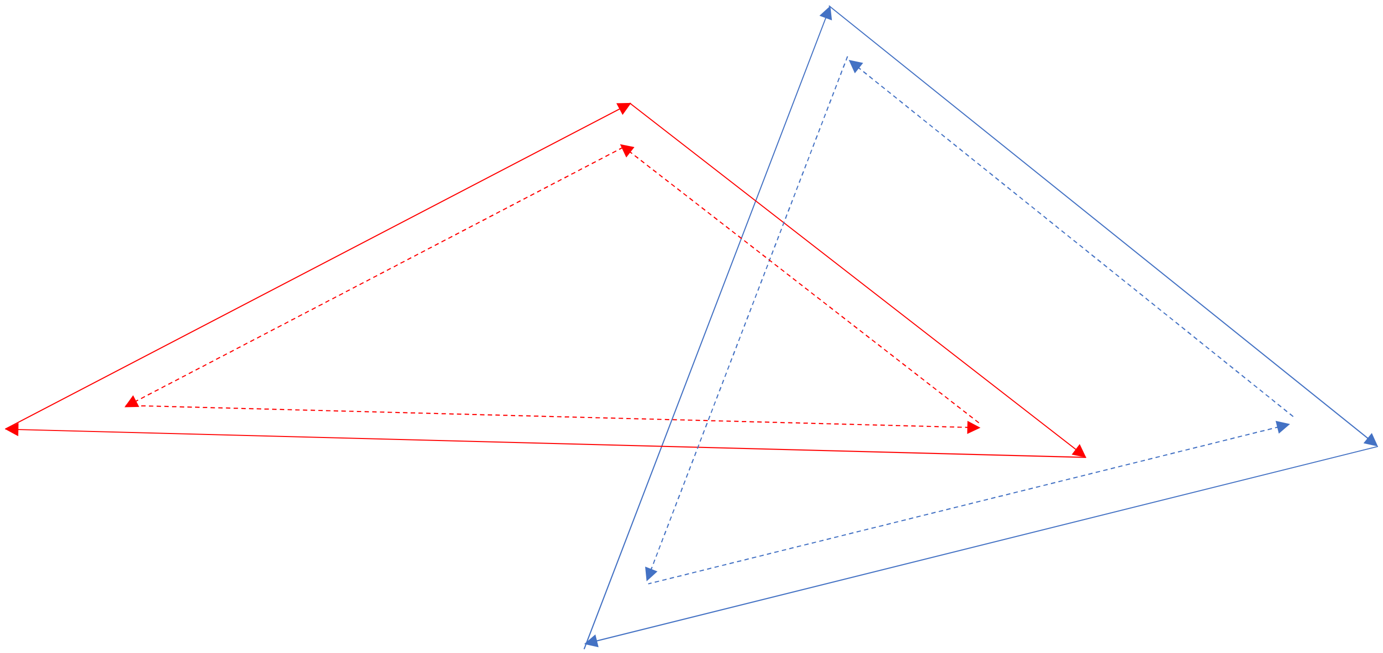
Example



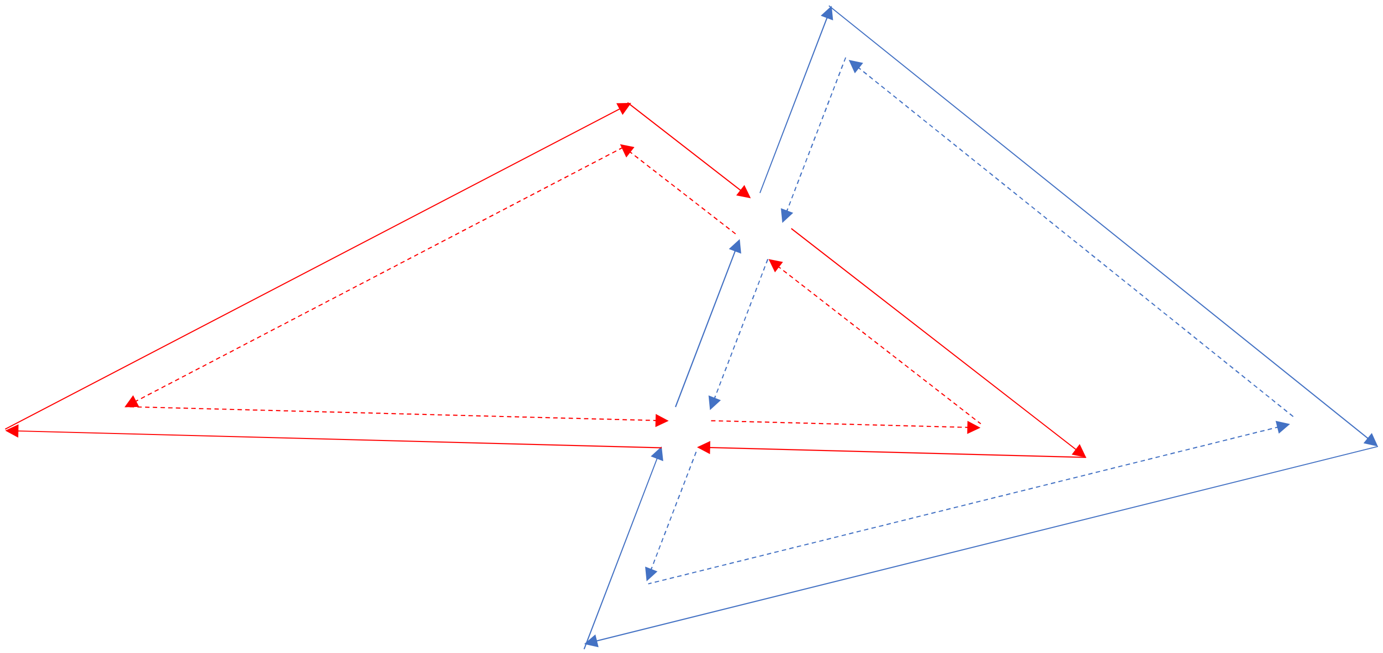
Clockwise Edges



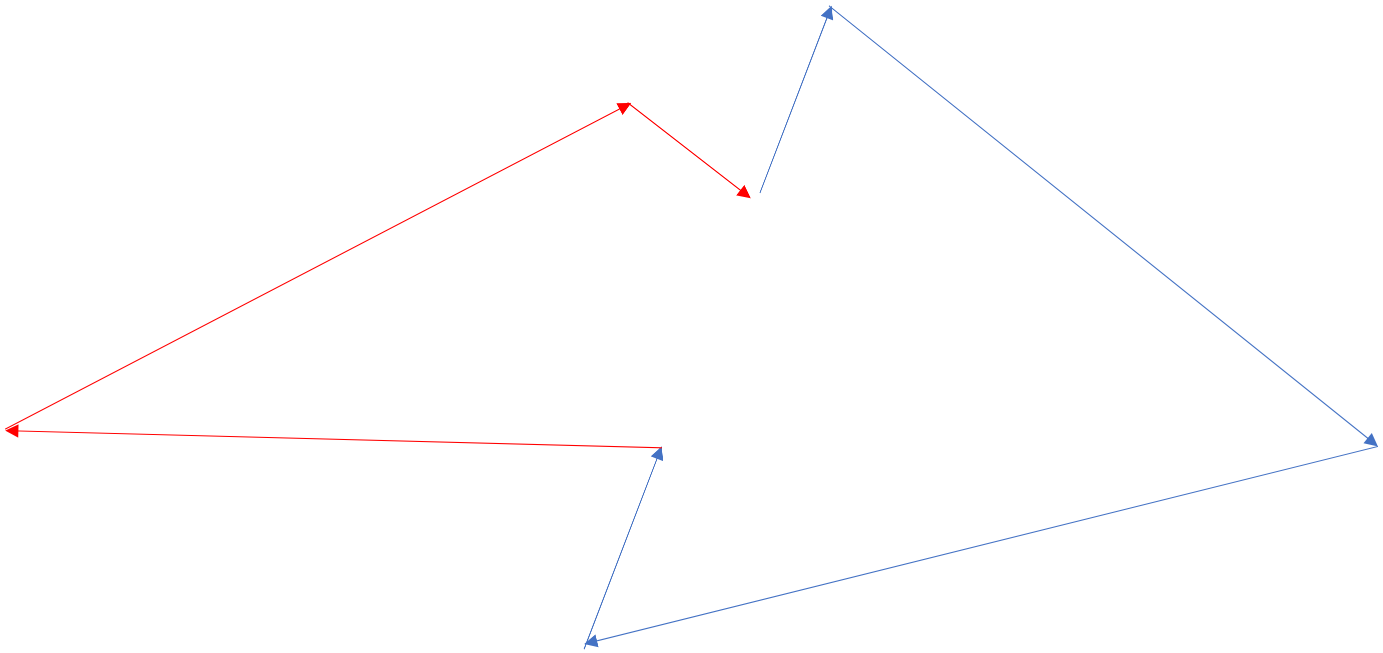
Outside and Inside Edges



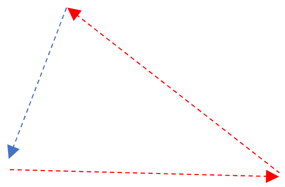
Intersections and Connections



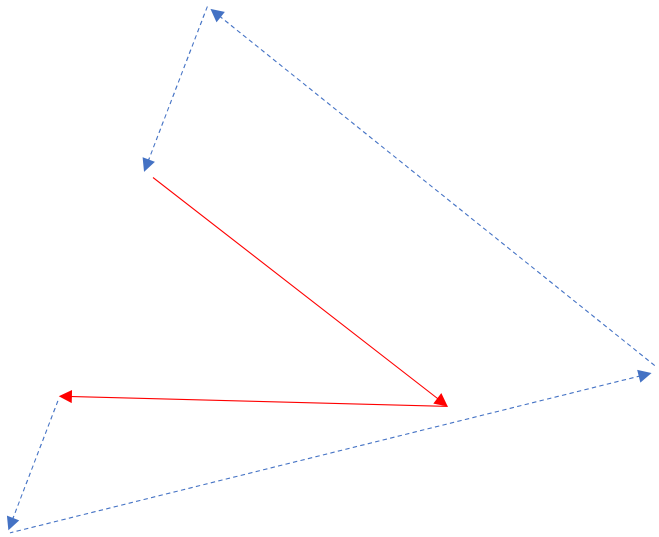
A U B



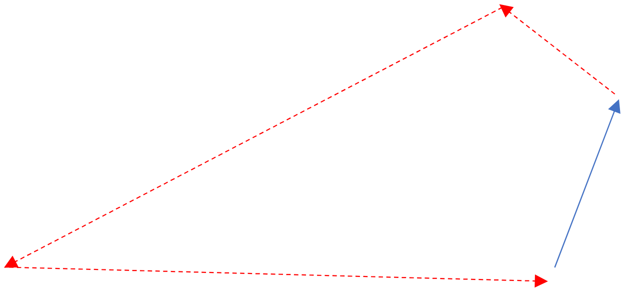
$A \cap B$



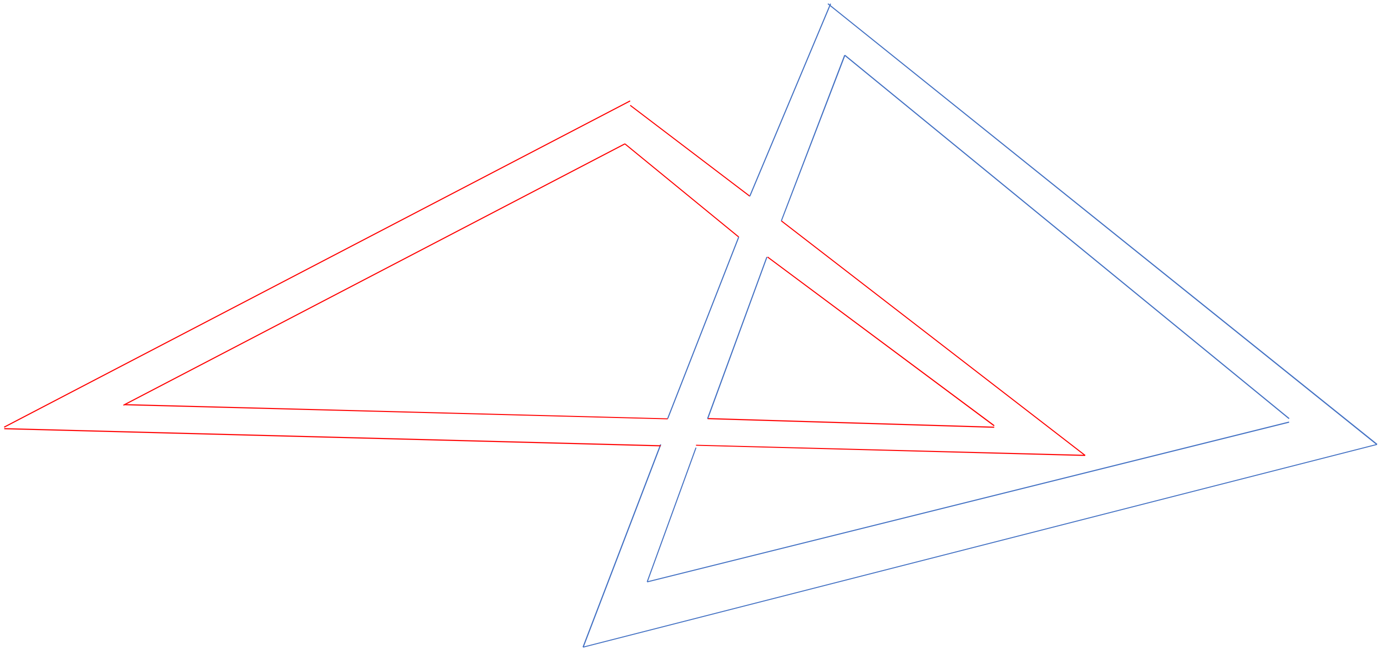
A - B



B - A



Intersections and Connections

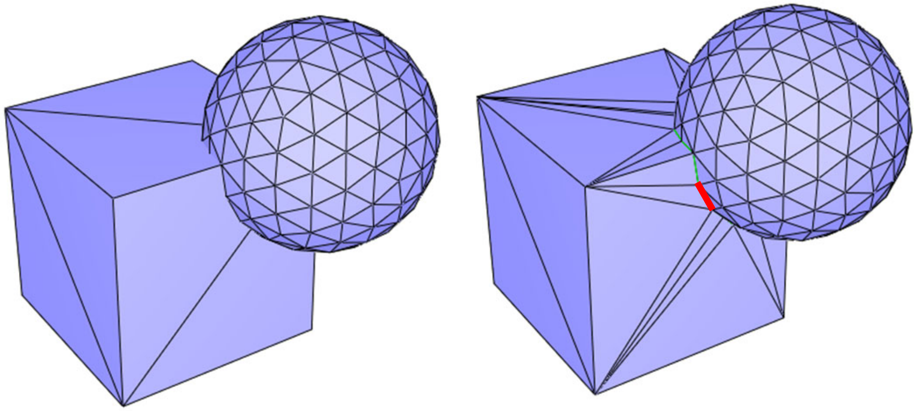


Weiler–Atherton Clipping Algorithm

- Compute the Boolean operations of two polygons
- Two 3D meshes?
- Same!

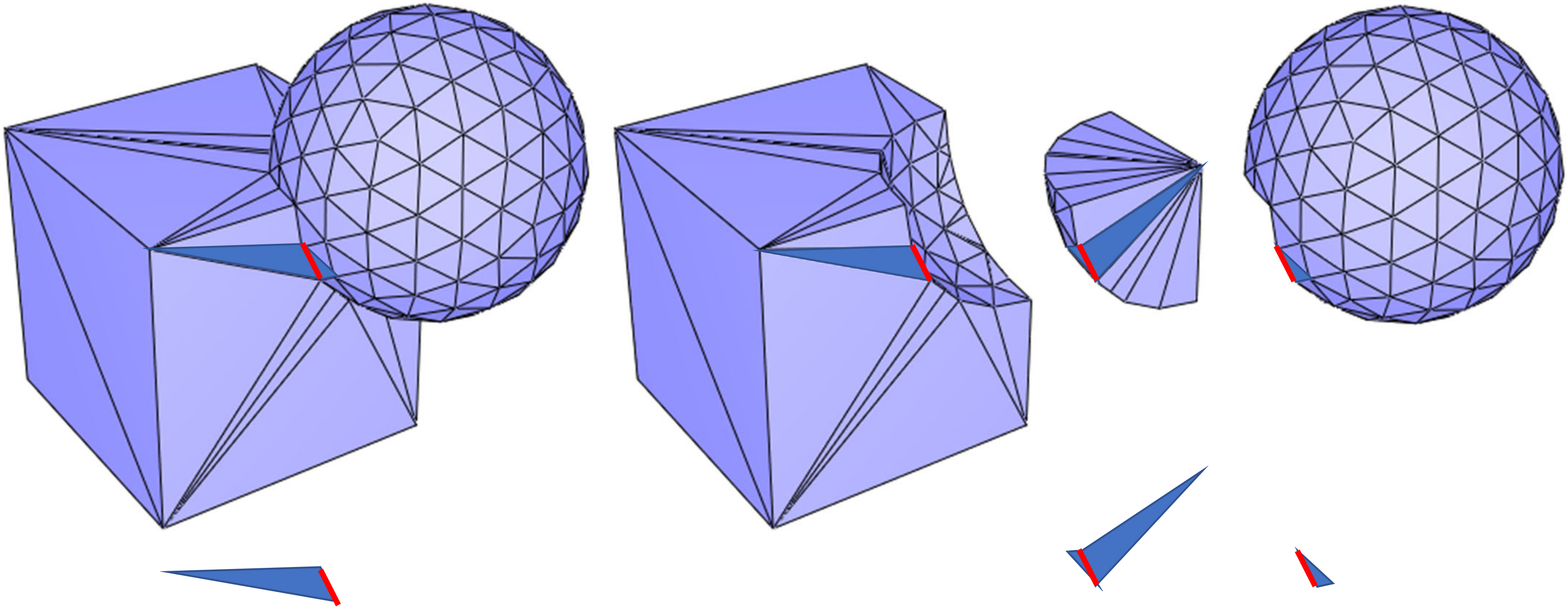
Weiler–Atherton Clipping Algorithm

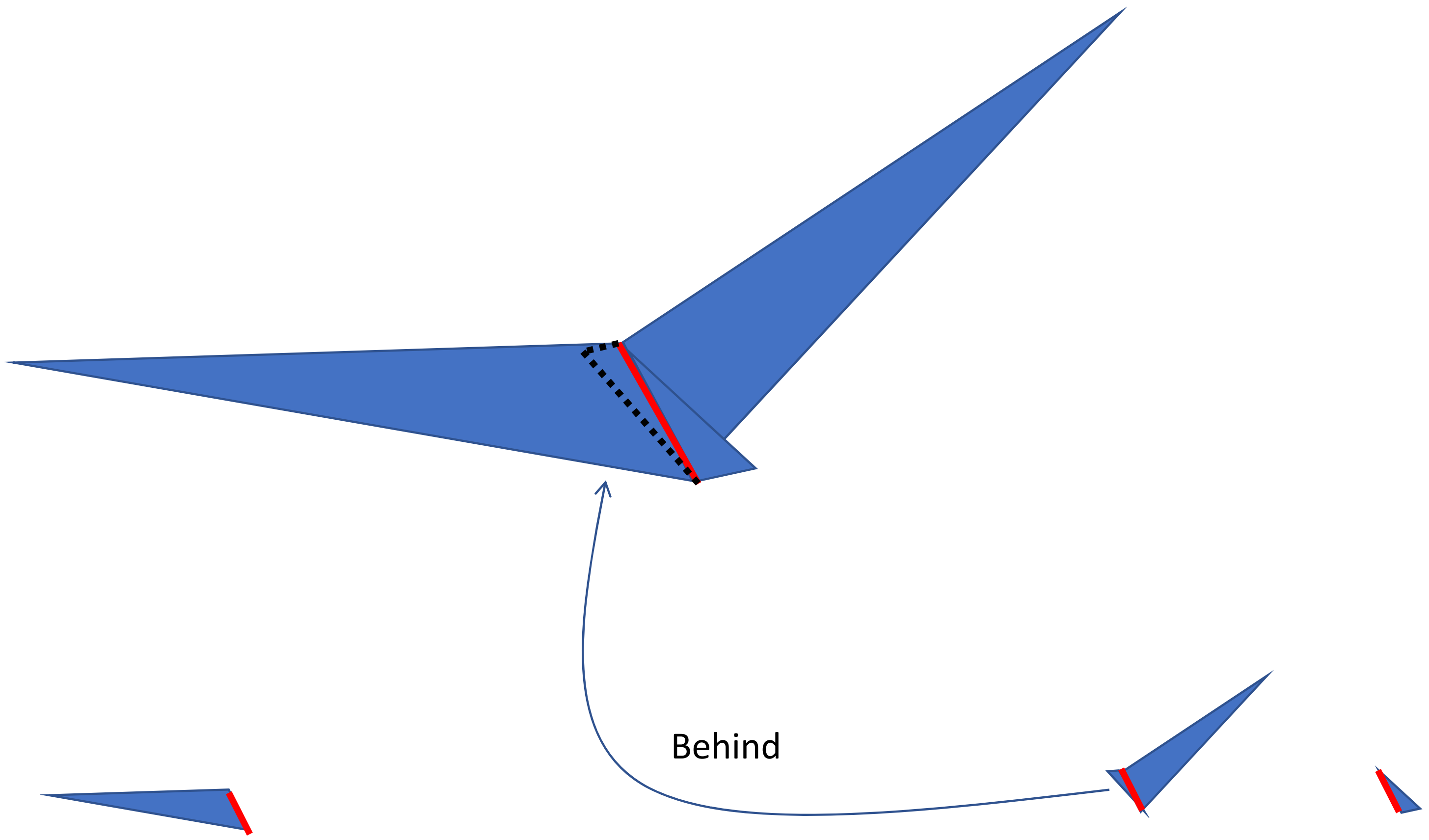
- Assume we can break every triangles into finer triangles and the two meshes only intersect in edges now
- For an edge in the intersection, there will be four triangles (from the two meshes) sharing it
- The crossing is exactly the same scenario of how we connect the edges in 2D



Weiler–Atherton Clipping Algorithm

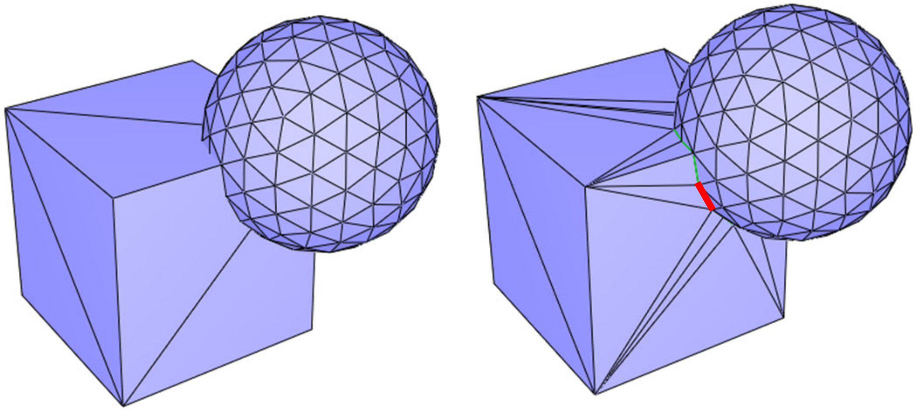
- Assume we can break every triangles into finer triangles and the two meshes only intersect in edges now
- For an edge in the intersection, there will be four triangles (from the two meshes) sharing it
- The crossing is exactly the same scenario of how we connect the edges in 2D




Behind

Wishful Thinking, HOW?

- Assume we can break every triangles into finer triangles and the two meshes only intersect in edges now
- For an edge in the intersection, there will be four triangles (from the two meshes) sharing it
- The crossing is exactly the same scenario of how we connect the edges in 2D

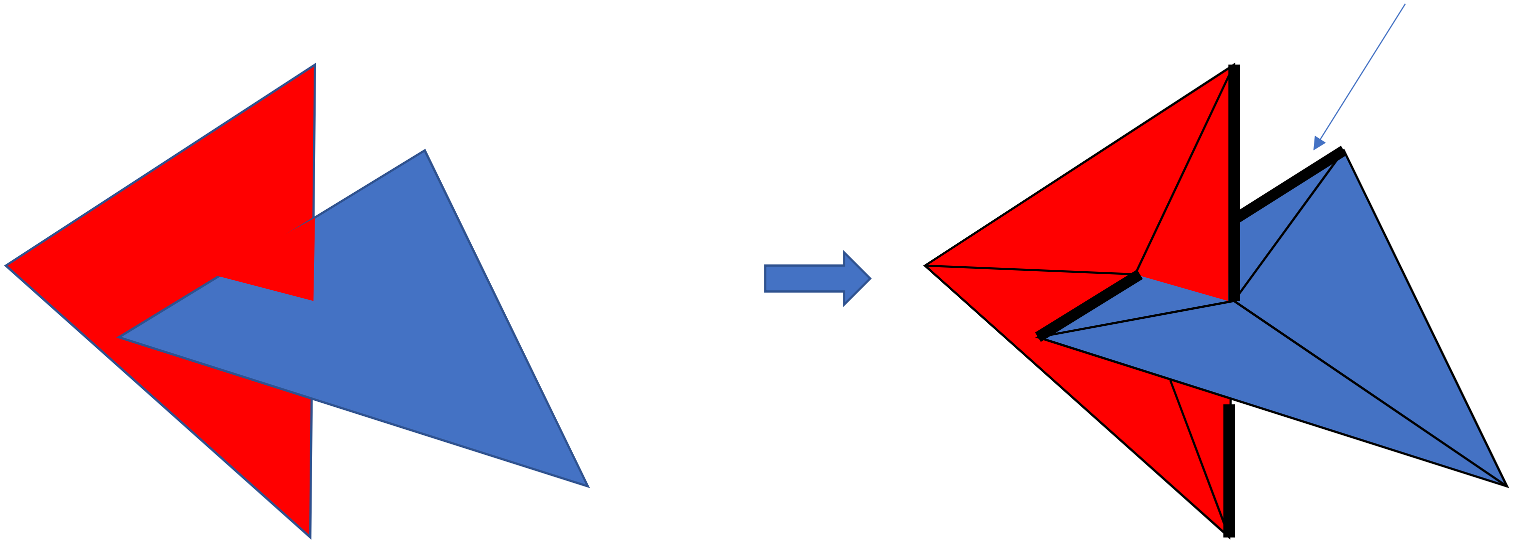


Wishful Thinking, HOW?

- Assume we can break every triangles into finer triangles and the two meshes only intersect in edges now
- Redefine “intersect”
  - Two triangle “*intersect*” edge other *iff* the intersection of their *interior* is non-empty
  - Interior of a triangle is the triangle minus its edges
- Put all triangles into a set
  - If a triangle does not intersect any other triangle, remove it from the set
  - Otherwise, break and retriangulates the two triangles and their neighbors
    - Replace all these new triangles in the set
  - Repeat until the set is empty

Down to Two Triangles Intersection

- How many types of intersection?
  - Leave it as an home exercise



And break the neighbors of these two edges

List of Final "Quests"

- Subdivision
  - Barycentric/Loop
  - Partial
  - Mesh Relaxation
- Self-intersection
  - Detection
  - Speed up
  - Object binary operations/Simple CSG
- Registrations
  - ICP
  - Speedup
  - P2P, P2S, P2?
  - Visualizing error between two models
- Decimation
  - Cluster/decimation
  - Quality control
  - Progressive mesh
- Thickening
  - With different cap types
  - Avoid self-intersection
- Remeshing
- Inspection
  - Given two similar meshes, calculate and visualize their errors
- Or any cool things you can think of

Grading Scheme

- Basic requirements (30%)
- UI (10%)
- Robustness/Extensiveness of test cases (20%)
- Enhancements/extensions (%40)
  - Features
  - Speedup