CSCI 4560/6560 Computational Geometry

https://www.cs.rpi.edu/~cutler/classes/computationalgeometry/S22/

Lecture 3: Map Overlay & Adjacency Data Structures

Outline for Today

- Questions about Homework 1?
  Questions about CGAL/Qt installation?
- Today's Motivation
- Minimal Representation (e.g., Essentially Data File Formats)
- Proper Data Structures w/ Adjacency
- Line Sweep Algorithm for Map Overlay
- Next Time

CGAL / Qt Installation Notes

HW1 autograding on Submitty not yet finished…

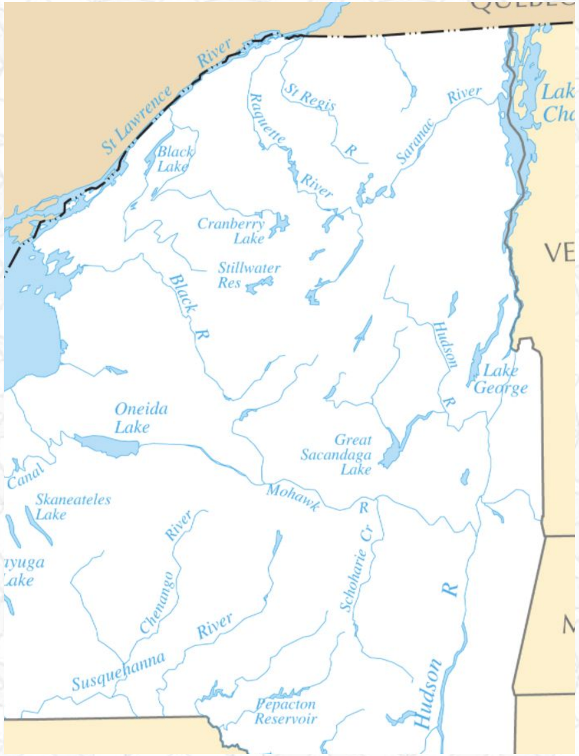
- Windows Notes
  - Make sure you're not using your WSL (Windows Subsystem for Linux / Ubuntu) or Cygwin terminals. *Why does Windows have so many terminals???*
  - You don't want to install the packages in WSL (`apt install …`)
  - You will use a Microsoft Visual Studio compiler. Not g++ or clang in WSL or Cygwin.
  - Be careful about 32 vs. 64 – either is fine! Just need to be consistent.
- Linux Notes
  - Check the version of CGAL. On Ubuntu 18.04 `apt install` only gets you CGAL 4.x All of the newer examples and documentation require CGAL 5.x – more work :(
  - You may need upgrade cmake – more work :(
- Mac Notes
  -

Outline for Today

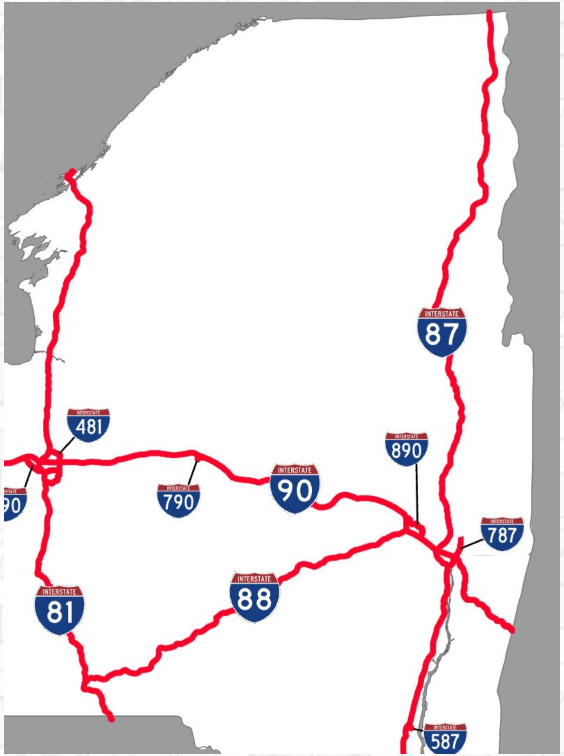
- Questions about Homework 1?
  Questions about CGAL/Qt installation?
- Today's Motivation
  - Problem Statement
  - Definition: Planar Subdivision
  - Euler's Formula
- Minimal Representation (e.g., Essentially Data File Formats)
- Proper Data Structures w/ Adjacency
- Line Sweep Algorithm for Map Overlay
- Next Time

Motivation for Last Lecture…

- 2 map layers storing the rivers & roads in NYS
- Each road/river stored as a *polyline - sequence of line segments*
- Find all intersections between a road segment and a river segment
- These are the bridges we need to build, inspect, repair, etc.



https://www.mapsof.net/new-york/new-york-rivers-and-lakes



https://upload.wikimedia.org/wikipedia/commons/1/17/NYInterstates.svg

Today's Motivation

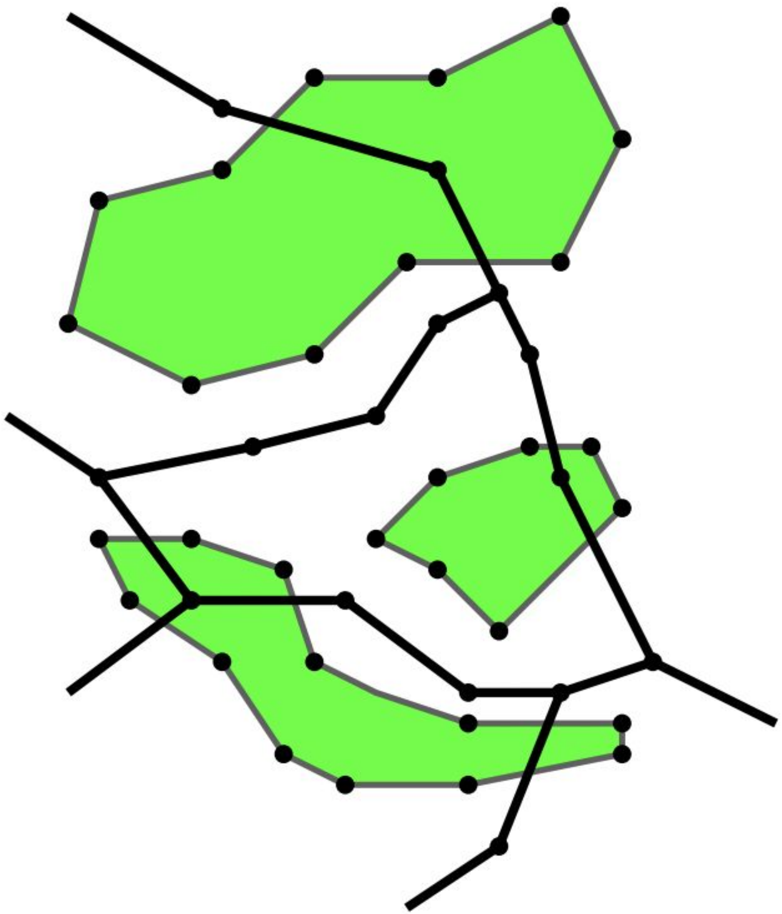
- Cartography (map making) is not just river and road polylines, it is also the areas or regions
- How do we describe and store a region?
- How do we overlay, intersect, & union map areas or regions?

https://www.natureconservancy.ca/assets/images/graphics/nat/maps/Forest-regions-map-NCC-1000px-custom.jpg

Today's Motivation

- "What is the total length of roads through forests?"

  → *Need to compute intersection of line segments with areas/regions.*

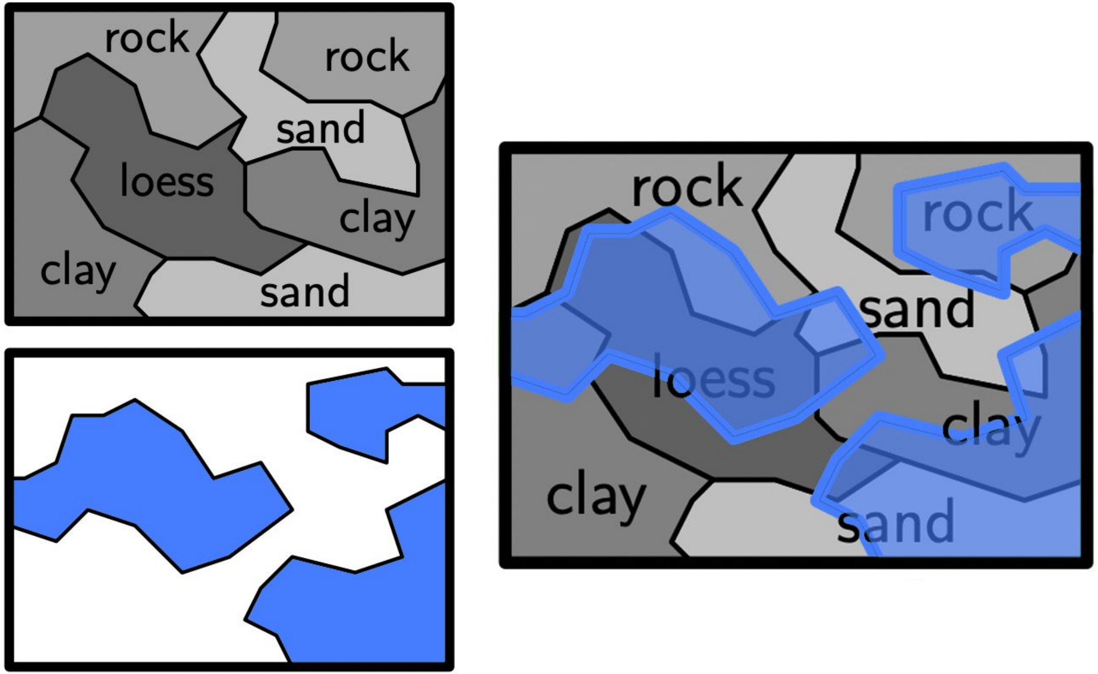


Frank Staals, http://www.cs.uu.nl/docs/vakken/ga/2021/

Today's Motivation

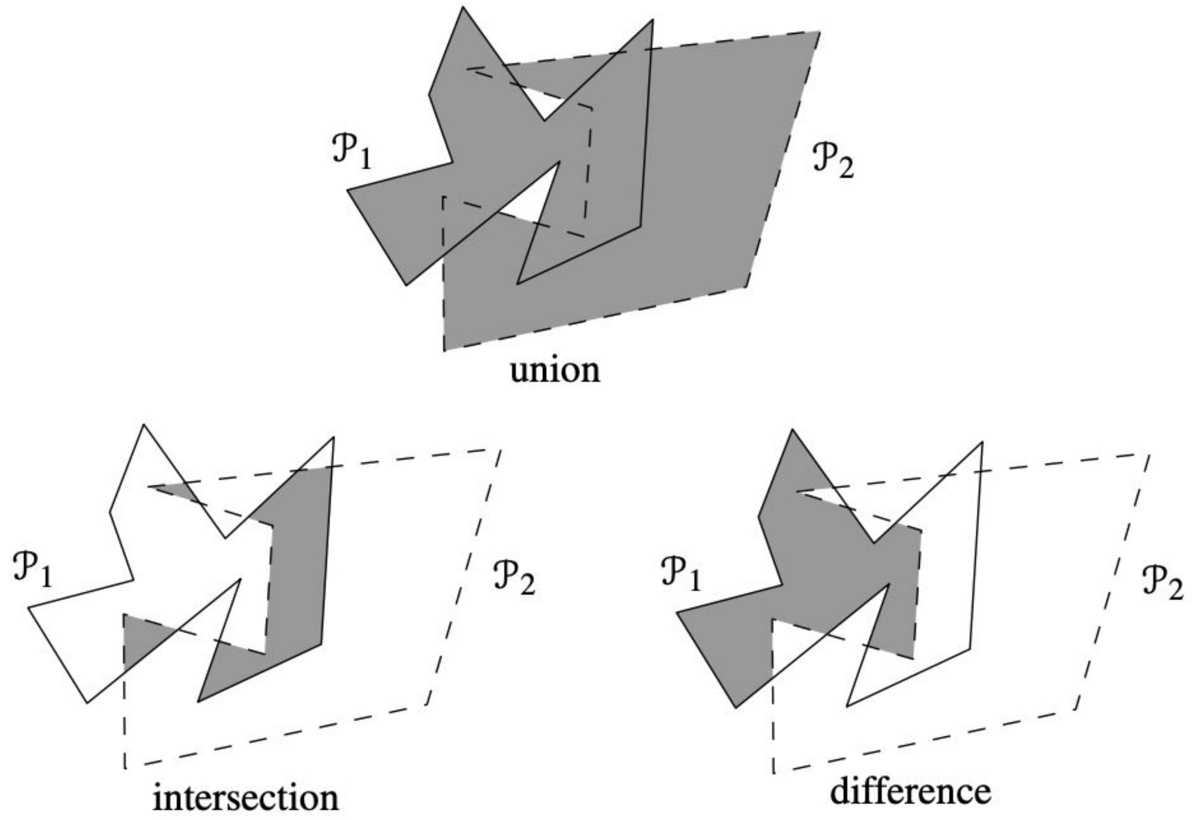
- "What is the total area of all lakes that occur over the geological soil type "rock"?

  → *Need to compute intersection* of areas/regions from two or more map layers



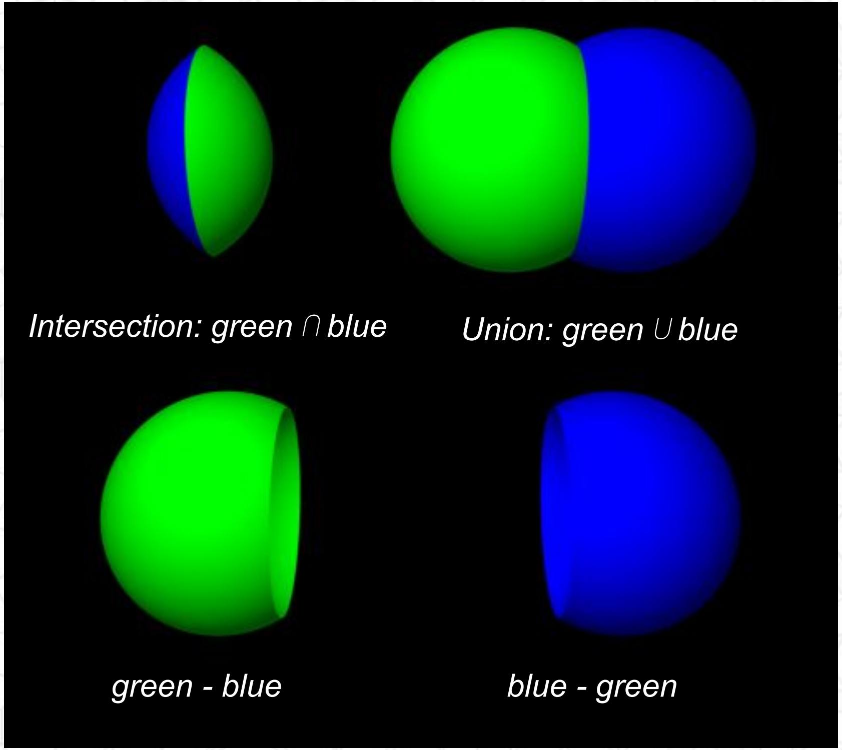
Frank Staals, http://www.cs.uu.nl/docs/vakken/ga/2021/

Boolean Operations

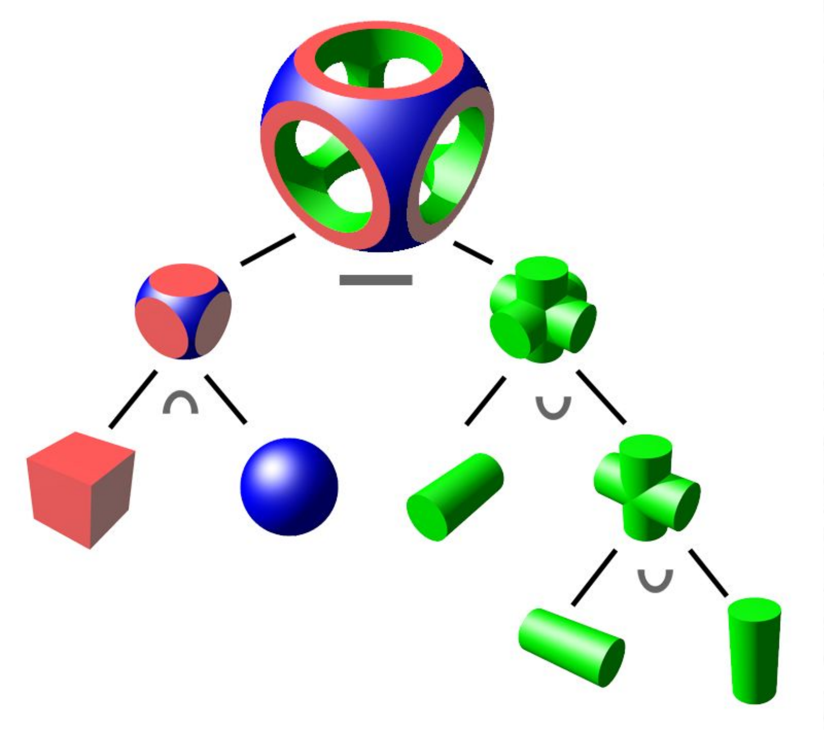


Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 2

CSG: Constructive Solid Geometry



http://matter.sawkmonkey.com/raytracer/csg.html



http://en.wikipedia.org/wiki/Constructive_solid_geometry#/media/File:Csg_tree.png

Outline for Today

- Questions about Homework 1?
  Questions about CGAL/Qt installation?
- Today's Motivation
  - Problem Statement
  - Definition: Planar Subdivision
  - Euler's Formula
- Minimal Representation (e.g., Essentially Data File Formats)
- Proper Data Structures w/ Adjacency
- Line Sweep Algorithm for Map Overlay
- Next Time

How to Represent Areas/Regions of a Plane?

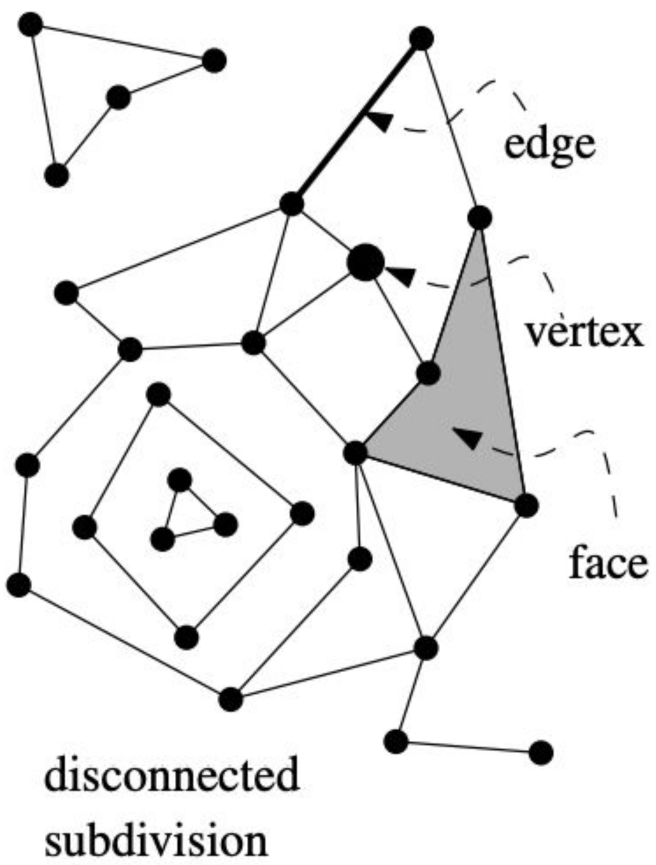
- A single map layer will label / subdivide the plane into *non-overlapping regions*
- The regions will be two-dimensional (planar)
- The regions may not be convex!
- The regions may have holes within them!
- Regions may be disconnected

https://www.natureconservancy.ca/assets/images/graphics/nat/maps/Forest-regions-map-NCC-1000px-custom.jpg

Planar Subdivision

- Edges are straight lines.
- An edge is “open” - it doesn’t include it’s endpoints.
- A face doesn’t include any points on its edges (or the vertices).
- Exactly one face, the “outer face”, is unbounded

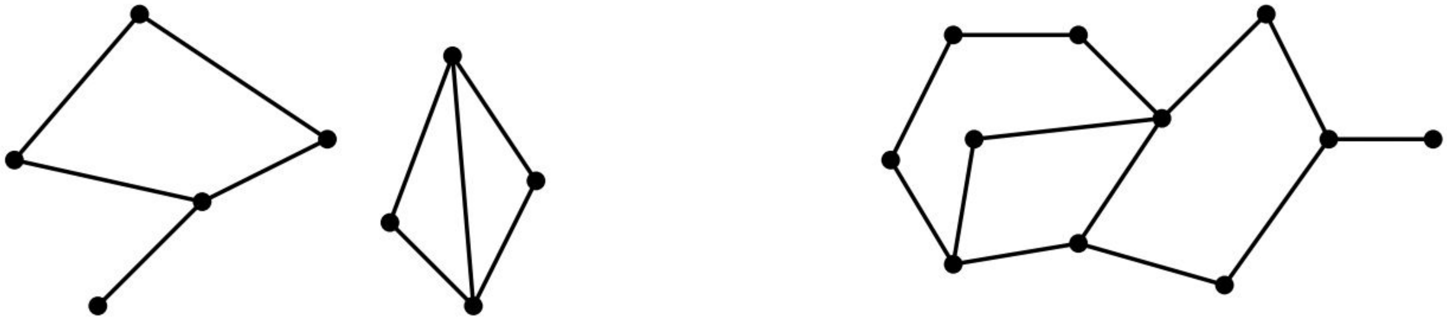
Every point in the plane is either a vertex, or on an edge, or on a face.



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 2

Euler's Formula for Planar Subdivision/Graph

For a planar, *connected* subdivision/graph
with V vertices, E edges, and F faces → $V - E + F = 2$ $V + F = E + 2$



$$V = 9,\ E = 10,\ F = 4$$

$$V - E + F = 3$$

$$V = 11,\ E = 13,\ F = 4$$

$$V - E + F = 2$$

$V - E + F > 2$ for unconnected an graph

Frank Staals, http://www.cs.uu.nl/docs/vakken/ga/2021/

Outline for Today

- Questions about Homework 1?
  Questions about CGAL/Qt installation?
- Today's Motivation
- Minimal Representation (e.g., Essentially Data File Formats)
  - List of Edges
  - List of Polygons
  - List of Unique Vertices & Indexed Faces
- Proper Data Structures w/ Adjacency
- Line Sweep Algorithm for Map Overlay
- Next Time

List of Edges:

Difficult Query:
How many faces are in this graph?

(3,6,2), (-6,2,4)

(2,2,4), (0,-1,-2)

(9,4,0), (4,2,9)

(8,8,7), (-4,-5,1)

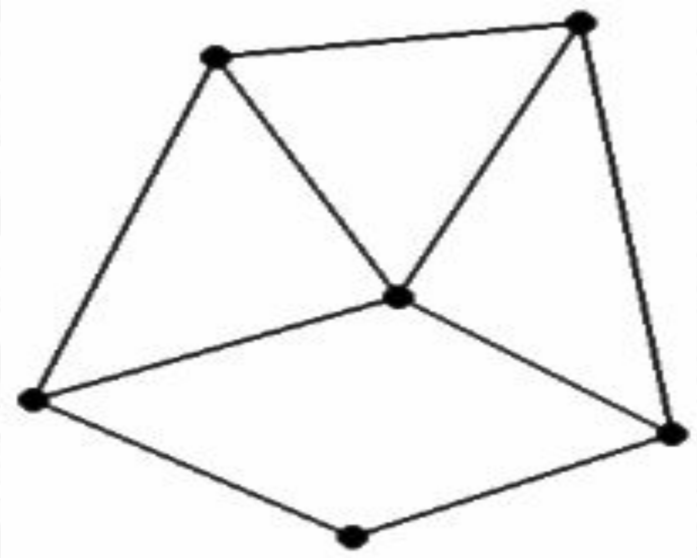
(-8,2,7), (1,2,-7)

(3,0,-3), (-7,4,-3)

(9,4,0), (4,2,9)

(3,6,2), (-6,2,4)

(-3,0,-4), (7,-3,-4)



List of Polygons:

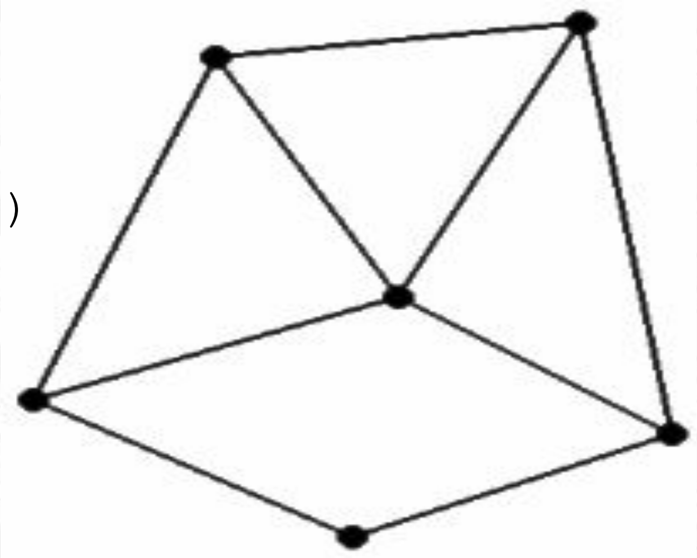
Expensive (& Not Robust) Query:
Which faces touch the quadrilateral face?

```
(3,-2,5), (3,6,2), (-6,2,4)

(2,2,4), (0,-1,-2), (9,4,0), (4,2,9)

(1,2,-2), (8,8,7), (-4,-5,1)

(-8,2,7), (-2,3,9), (1,2,-7)
```



List of Unique Vertices & Indexed Faces:

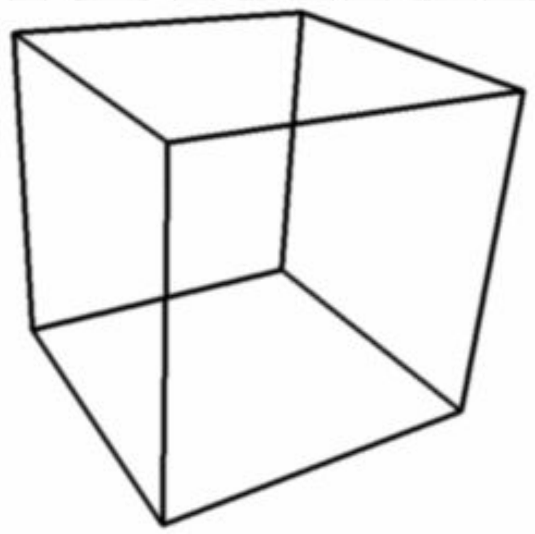
Vertices:

```
(-1, -1, -1)
(-1, -1, 1)
(-1, 1, -1)
(-1, 1, 1)
(1, -1, -1)
(1, -1, 1)
(1, 1, -1)
(1, 1, 1)
```

Faces:

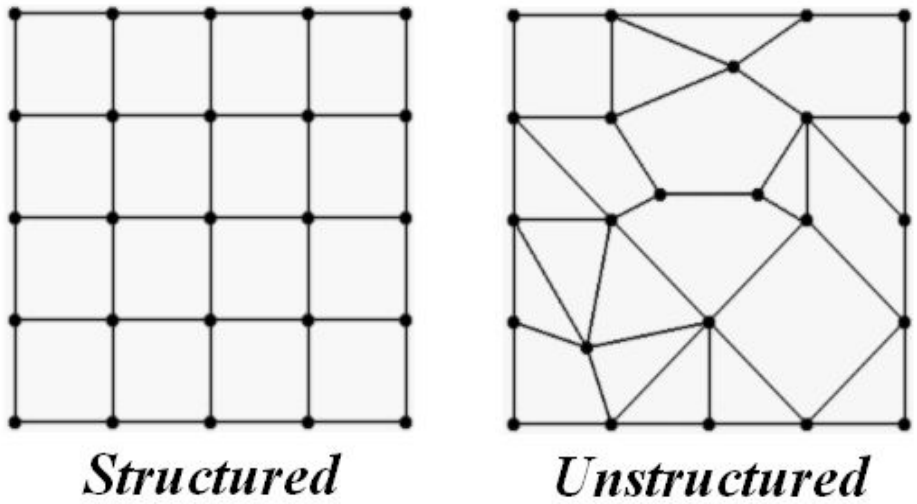
```
1 2 4 3
5 7 8 6
1 5 6 2
3 4 8 7
1 3 7 5
2 6 8 4
```

Expensive Query:
Which faces use the upper left vertex?



Problems with Simple Lists

- No Neighbor / Adjacency Information
- Linear-time Searches



- Adjacency is implicit for structured meshes, but what do we do for unstructured meshes?

Outline for Today

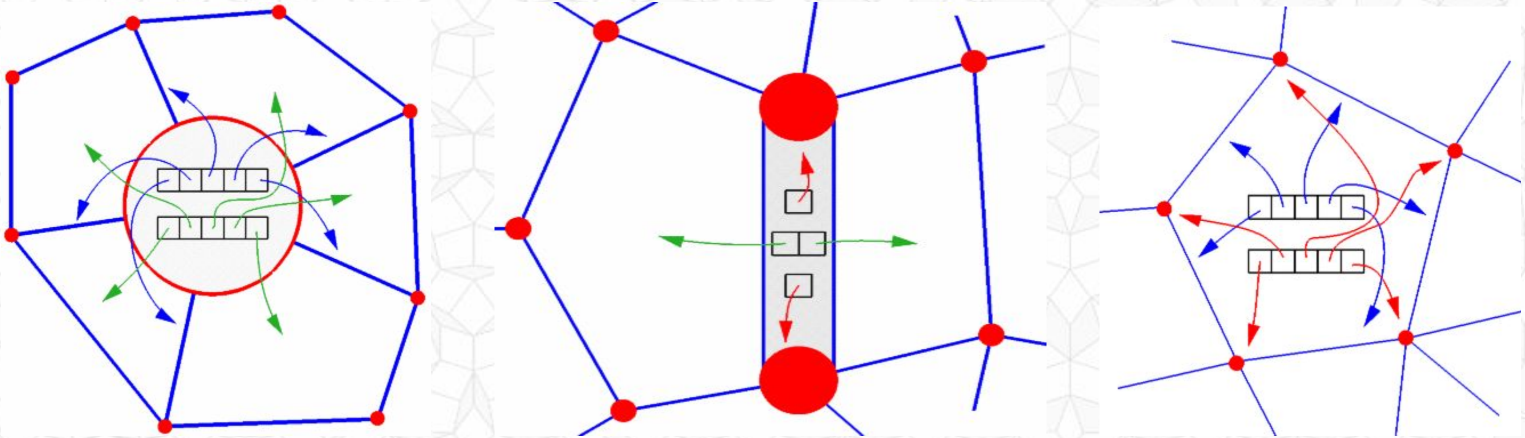
- Questions about Homework 1?
  Questions about CGAL/Qt installation?
- Today's Motivation
- Minimal Representation (e.g., Essentially Data File Formats)
- Proper Data Structures w/ Adjacency
  - Simple / Exhaustive Adjacency
  - Fixed Storage Data Structures - Winged Edge
  - Fixed Computation Data Structures - Half Edge
- Line Sweep Algorithm for Map Overlay
- Next Time

Mesh Data

- So, in addition to:
  - Geometric Information (position)
  - Attribute Information (color, texture, temperature, population density, etc.)
- Let's store:
  - Topological Information (adjacency, connectivity)

Simple / Exhaustive Adjacency

- Each element (vertex, edge, and face) has a list of pointers to all incident elements
- Queries depend only on local complexity of mesh
- Data structures do not have fixed size
- Slow! Big! Too much work to maintain!



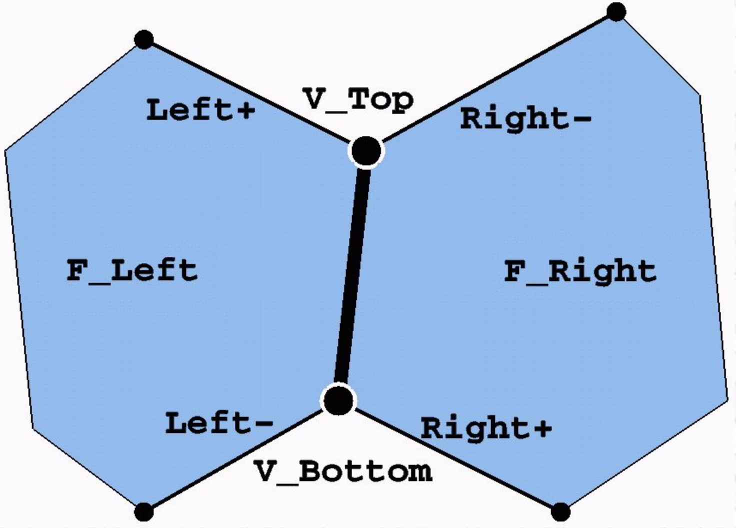
Original slide from
Justin Legakis

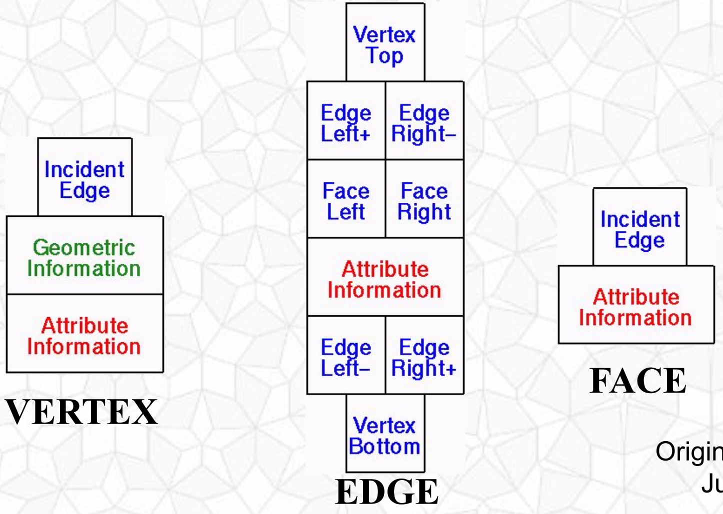
Outline for Today

- Questions about Homework 1?
  Questions about CGAL/Qt installation?
- Today's Motivation
- Minimal Representation (e.g., Essentially Data File Formats)
- Proper Data Structures w/ Adjacency
  - Simple / Exhaustive Adjacency
  - Fixed Storage Data Structures - Winged Edge
  - Fixed Computation Data Structures - Half Edge
- Line Sweep Algorithm for Map Overlay
- Next Time

Winged Edge (Baumgart, 1975)

- Edges will store everything!
- Vertices and Faces will point to an edge
- Data Structure Size?
- How do we gather all faces surrounding one vertex?

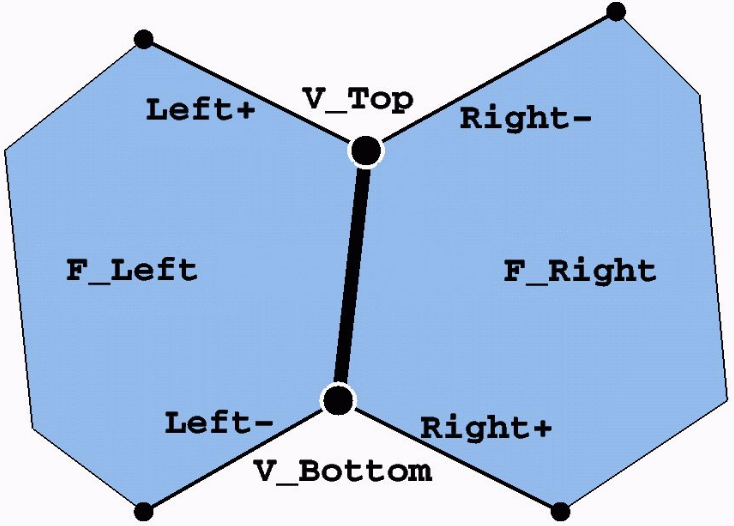


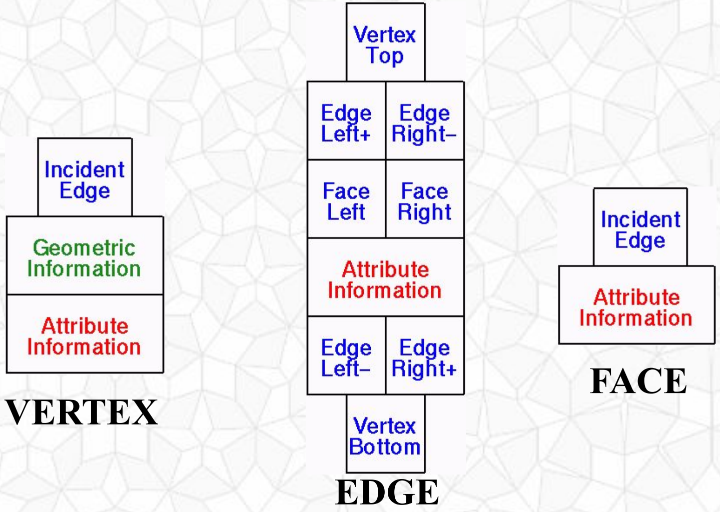


Original slide from Justin Legakis

Winged Edge (Baumgart, 1975)

- Edges will store everything!
- Vertices and Faces will point to an edge
- Data Structure Size?
  Fixed
- How do we gather all faces surrounding one vertex?
  Messy, because there is no CONSISTENT way to order pointers!

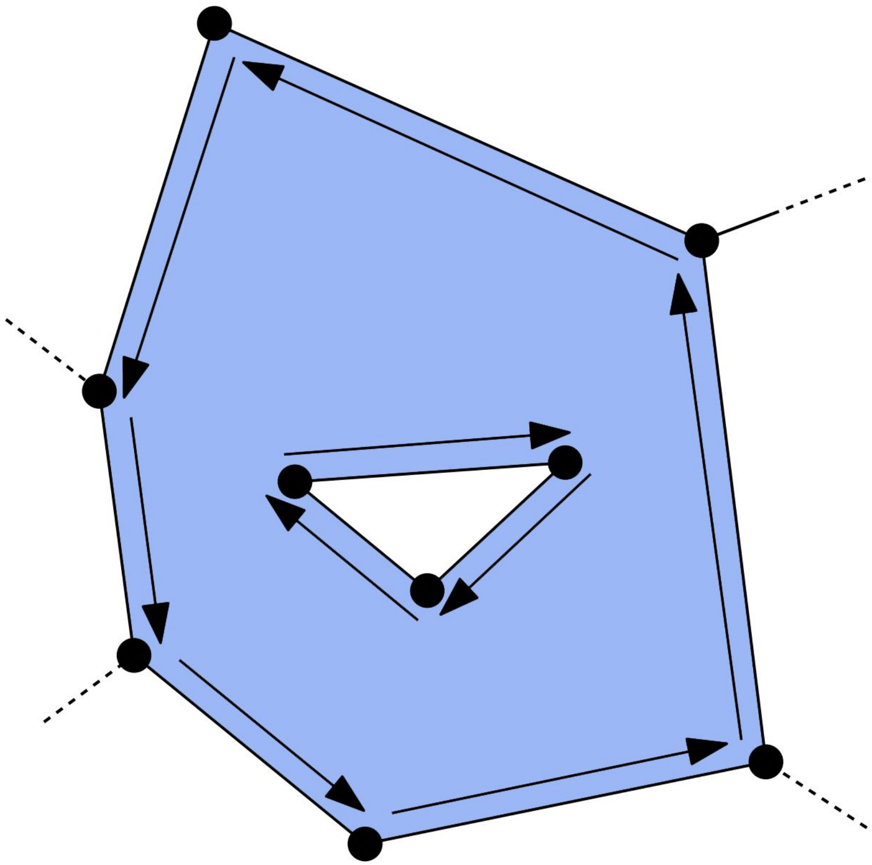




Consistent Edge Orientation

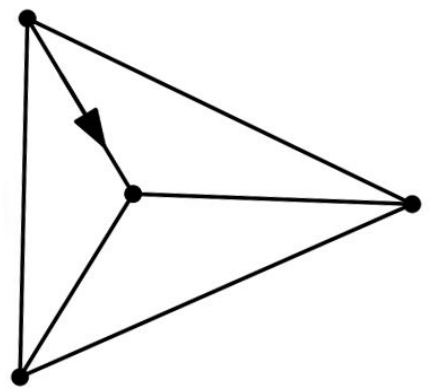
- It is desirable to have a consistent orientation for edges that define the boundary of a region / face.
- This will clearly indicate which points are inside/on the face.
- Especially if the face has one or more interior holes.

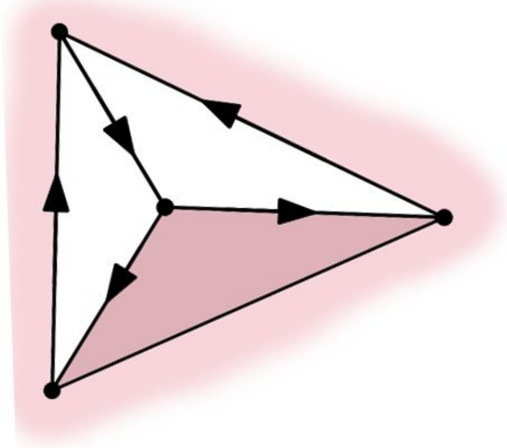
Counter-clockwise in this image… but don't be surprised to see different standards…



Consistent Edge Orientation

- It would be useful to have a consistent orientation *(clockwise or counterclockwise)* for all edges that define the boundary of a region / face.
- This will simplify traversal around the boundary – reducing if/else branches, etc.
- However, most meshes cannot be labeled such that the edges of every face are consistently oriented.



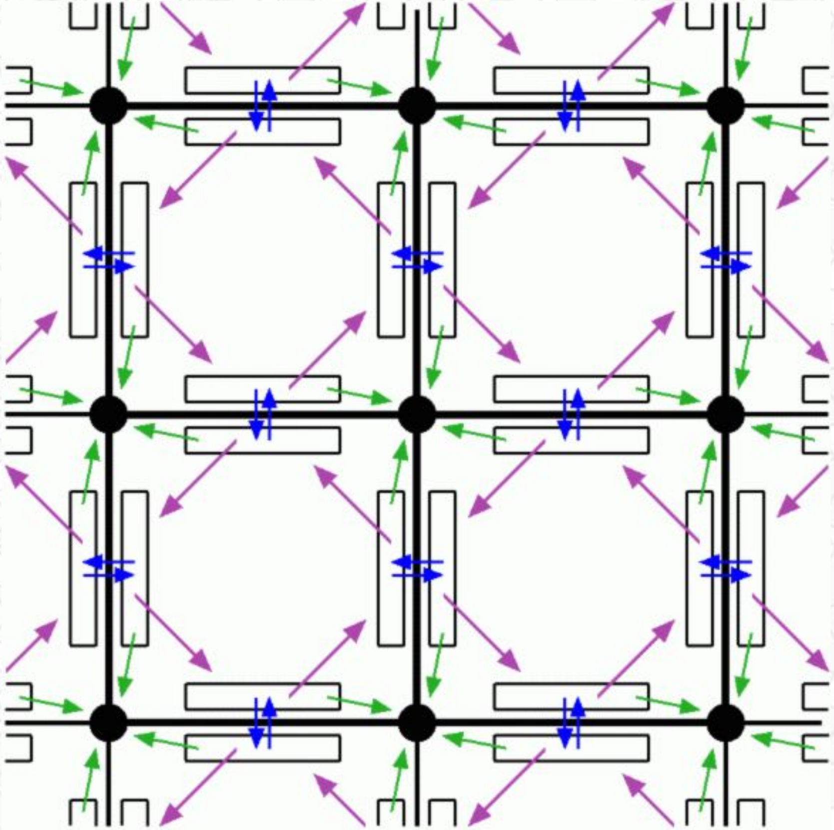


Outline for Today

- Questions about Homework 1?
  Questions about CGAL/Qt installation?
- Today’s Motivation
- Minimal Representation (e.g., Essentially Data File Formats)
- Proper Data Structures w/ Adjacency
  - Simple / Exhaustive Adjacency
  - Fixed Storage Data Structures - Winged Edge
  - Fixed Computation Data Structures - Half Edge
- Line Sweep Algorithm for Map Overlay
- Next Time

HalfEdge (Eastman, 1982)

- Every edge is represented by two directed HalfEdge structures
- Each HalfEdge stores:
  - **vertex** at end of directed edge
  - **symmetric** half edge
  - **face** to left of edge
  - **next** points to the HalfEdge counter-clockwise around face on left
- Orientation is essential, but can be done consistently!



Original slide from Justin Legakis

HalfEdge (Eastman, 1982)

- Starting at a half edge, how do we find:

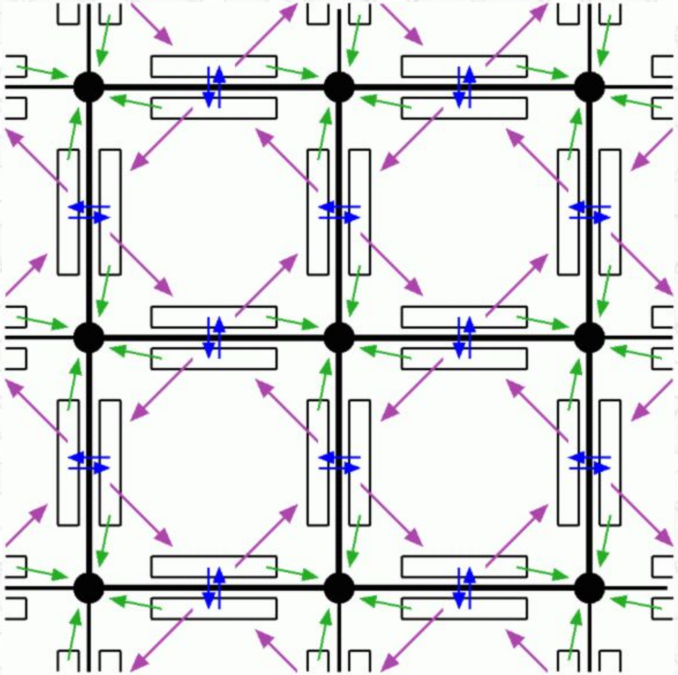
  the other vertex of the edge?

  the other face of the edge?

  the clockwise edge around
  the face at the left?

  all the edges surrounding
  the face at the left?

  all the faces surrounding
  the vertex?



Original slide from Justin Legakis

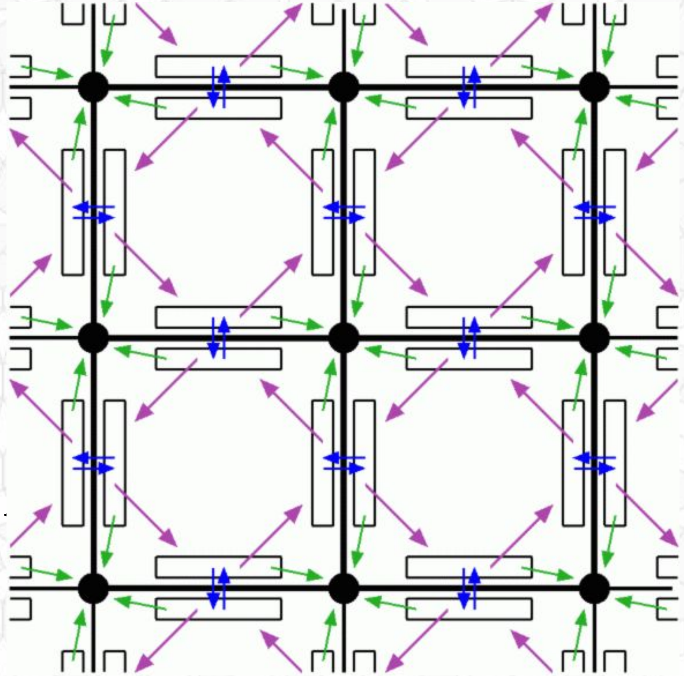
HalfEdge (Eastman, 1982)

- Loop around a Face:

```
HalfEdgeMesh::FaceLoop(HalfEdge *HE) {
    HalfEdge *loop = HE;
    do {
        loop = loop->Next;
    } while (loop != HE);
}
```

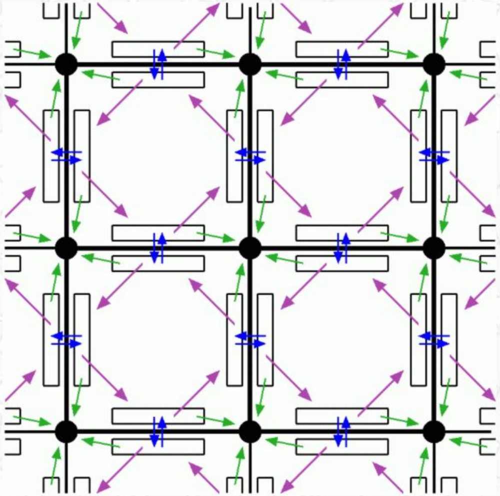
- Loop around a Vertex:

```
HalfEdgeMesh::VertexLoop(HalfEdge *HE) {
    HalfEdge *loop = HE;
    do {
        loop = loop->Next->Sym;
    } while (loop != HE);
}
```



Original slide from Justin Legakis

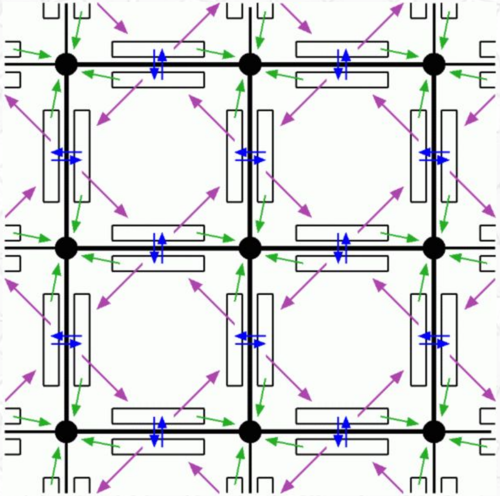
HalfEdge (Eastman, 1982)



- Data Structure Size?

- Data:
  - geometric information stored at Vertices
  - attribute information in Vertices, HalfEdges, and/or Faces
  - topological information in HalfEdges only!
- Orientable surfaces only (no Mobius Strips!)
- Local consistency everywhere implies global consistency
- Time Complexity?

HalfEdge (Eastman, 1982)



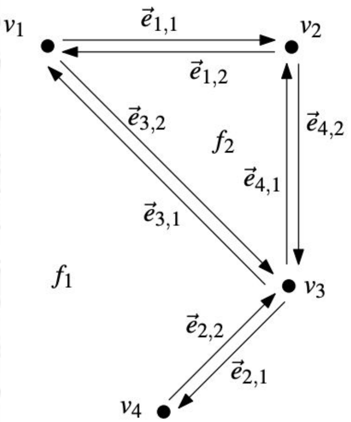
- Data Structure Size?
  Fixed
- Data:
  - geometric information stored at Vertices
  - attribute information in Vertices, HalfEdges, and/or Faces
  - topological information in HalfEdges only!
- Orientable surfaces only (no Mobius Strips!)
- Local consistency everywhere implies global consistency
- Time Complexity?
  linear in the amount of information gathered

HalfEdge (Eastman, 1982)

- Data Structure Size?
  Fixed

  … Unless interior holes are allowed – then faces will need to store a list of one edge for each hole.

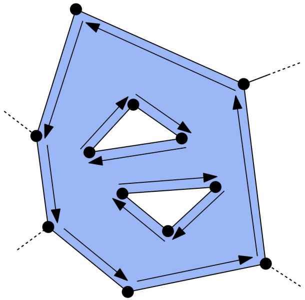
Could be a list of arbitrary length!



Vertex	Coordinates	IncidentEdge
v_1	$(0,4)$	$\vec{e}_{1,1}$
v_2	$(2,4)$	$\vec{e}_{4,2}$
v_3	$(2,2)$	$\vec{e}_{2,1}$
v_4	$(1,1)$	$\vec{e}_{2,2}$

Face	OuterComponent	InnerComponents
f_1	**nil**	$\vec{e}_{1,1}$
f_2	$\vec{e}_{4,1}$	**nil**

Half-edge	Origin	Twin	IncidentFace	Next	Prev
$\vec{e}_{1,1}$	v_1	$\vec{e}_{1,2}$	f_1	$\vec{e}_{4,2}$	$\vec{e}_{3,1}$
$\vec{e}_{1,2}$	v_2	$\vec{e}_{1,1}$	f_2	$\vec{e}_{3,2}$	$\vec{e}_{4,1}$
$\vec{e}_{2,1}$	v_3	$\vec{e}_{2,2}$	f_1	$\vec{e}_{2,2}$	$\vec{e}_{4,2}$
$\vec{e}_{2,2}$	v_4	$\vec{e}_{2,1}$	f_1	$\vec{e}_{3,1}$	$\vec{e}_{2,1}$
$\vec{e}_{3,1}$	v_3	$\vec{e}_{3,2}$	f_1	$\vec{e}_{1,1}$	$\vec{e}_{2,2}$
$\vec{e}_{3,2}$	v_1	$\vec{e}_{3,1}$	f_2	$\vec{e}_{4,1}$	$\vec{e}_{1,2}$
$\vec{e}_{4,1}$	v_3	$\vec{e}_{4,2}$	f_2	$\vec{e}_{1,2}$	$\vec{e}_{3,2}$
$\vec{e}_{4,2}$	v_2	$\vec{e}_{4,1}$	f_1	$\vec{e}_{2,1}$	$\vec{e}_{1,1}$



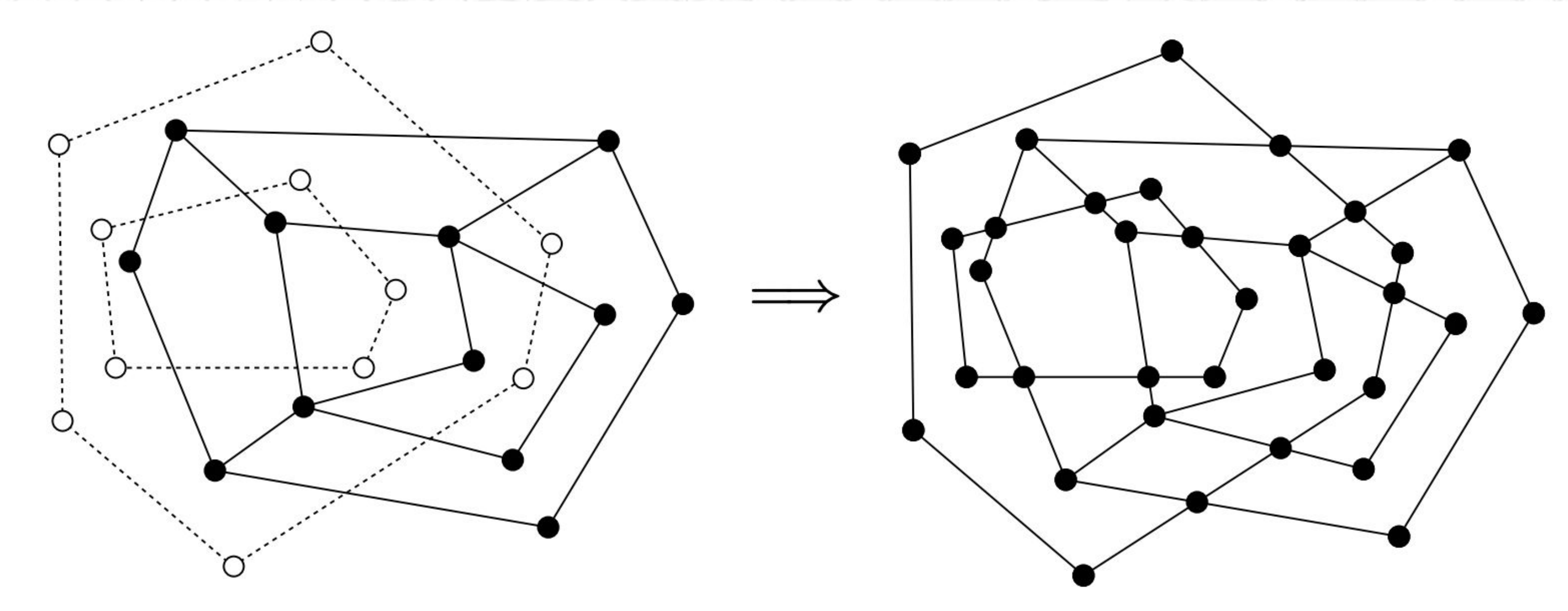
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 2

Outline for Today

- Questions about Homework 1?
  Questions about CGAL/Qt installation?
- Today's Motivation
- Minimal Representation (e.g., Essentially Data File Formats)
- Proper Data Structures w/ Adjacency
- Line Sweep Algorithm for Map Overlay
  - Enumerate Intersection Cases for Map Overlay
  - Update Edges, Vertices, and Faces
  - Analysis
- Next Time

Input: Doubly-connected, half-edge repr. for planar subdivisions, S_1 and S_2

Output: Doubly-connected, half-edge repr. for overlay subdivision $O(S_1,S_2)$.

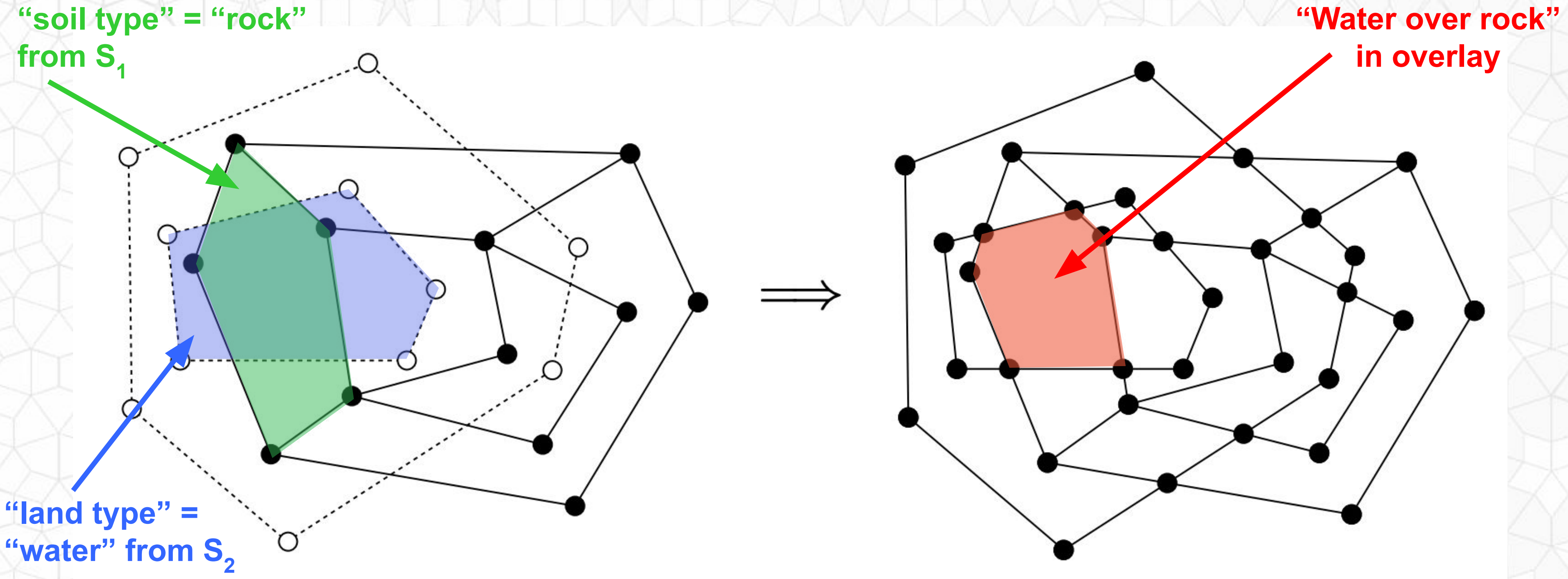


Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 2

Input: Doubly-connected, half-edge repr. for planar subdivisions, S_1 and S_2

Output: Doubly-connected, half-edge repr. for overlay subdivision $O(S_1,S_2)$.

Every face in overlay is labeled with the attribute info from a face from S_1 and S_2.

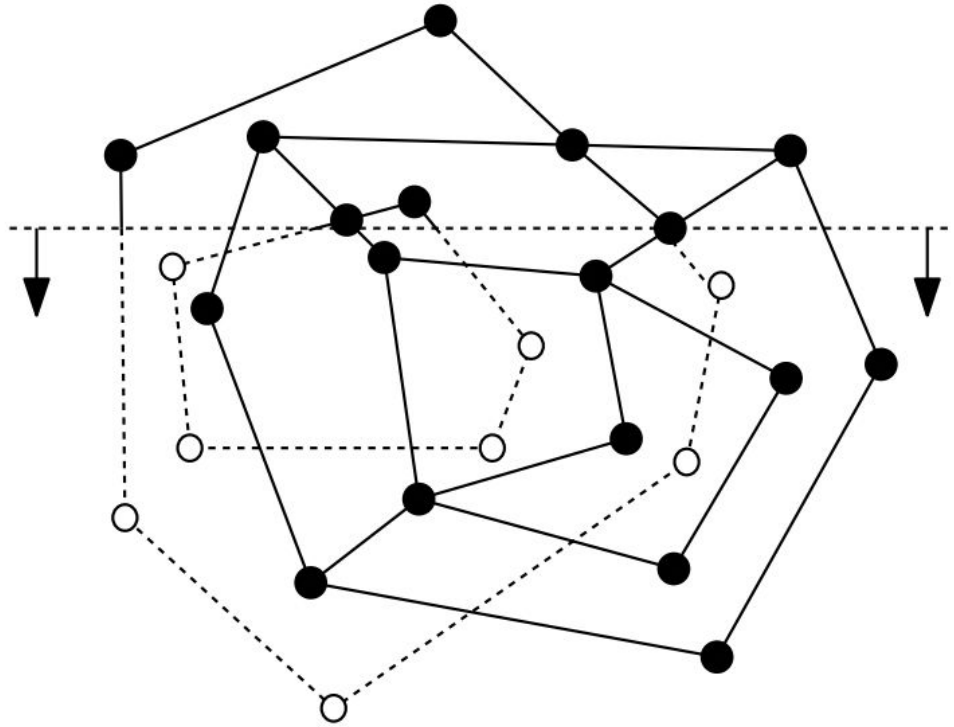


Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 2

- Step 1: Copy all of the half edges from both S_1 and S_2 to new structure D.

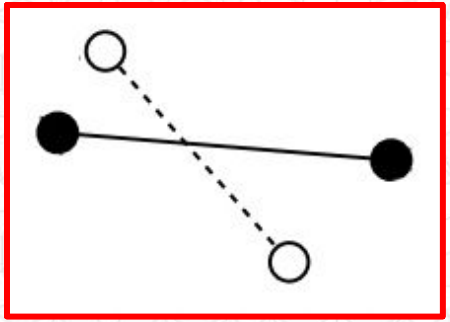
- Step 2: Perform the line sweep edge intersection algorithm from Lecture 2 to identify intersections between a segment in S_1 and a segment in S_2

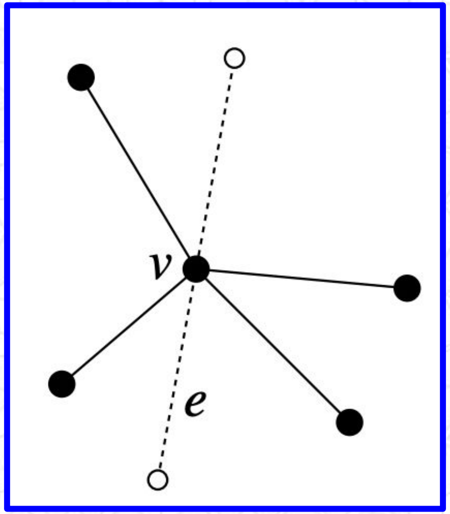
  These edges in D will need to be edited - cut at the intersection point - new edges will need to be added. Also new vertices and new Face edits/additions.



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 2

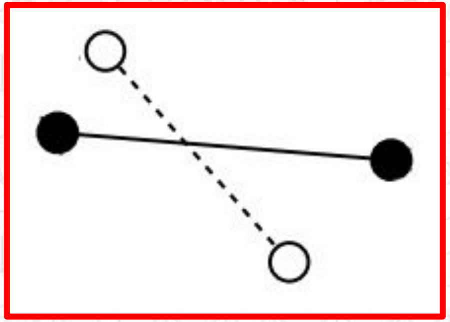
Events that will be encountered during Line Sweep

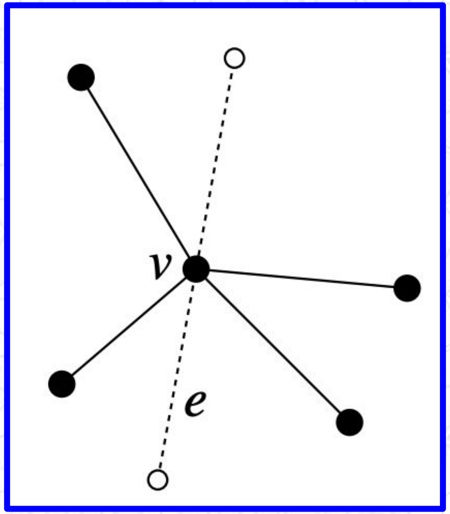




Events that will be encountered during Line Sweep

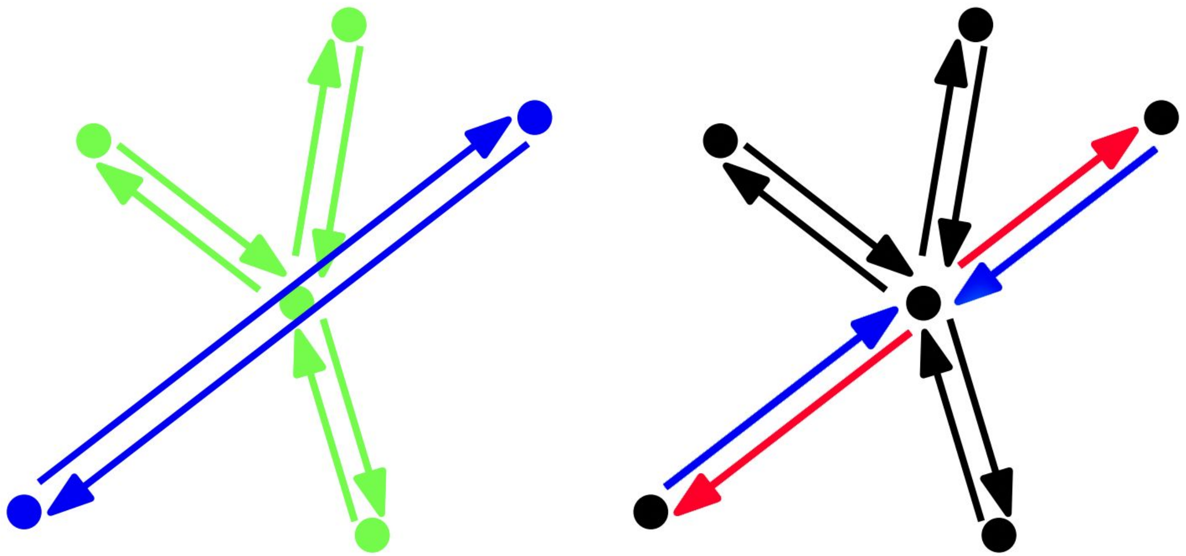
- A vertex in S_1
- A vertex in S_2
- Intersection between edge in S_1 and edge in S_2
- Intersection between vertex in S_1 and edge in S_2
- Intersection between edge in S_1 and vertex in S_2
- Intersection between vertex in S_1 and vertex in S_2





Must handle each case…

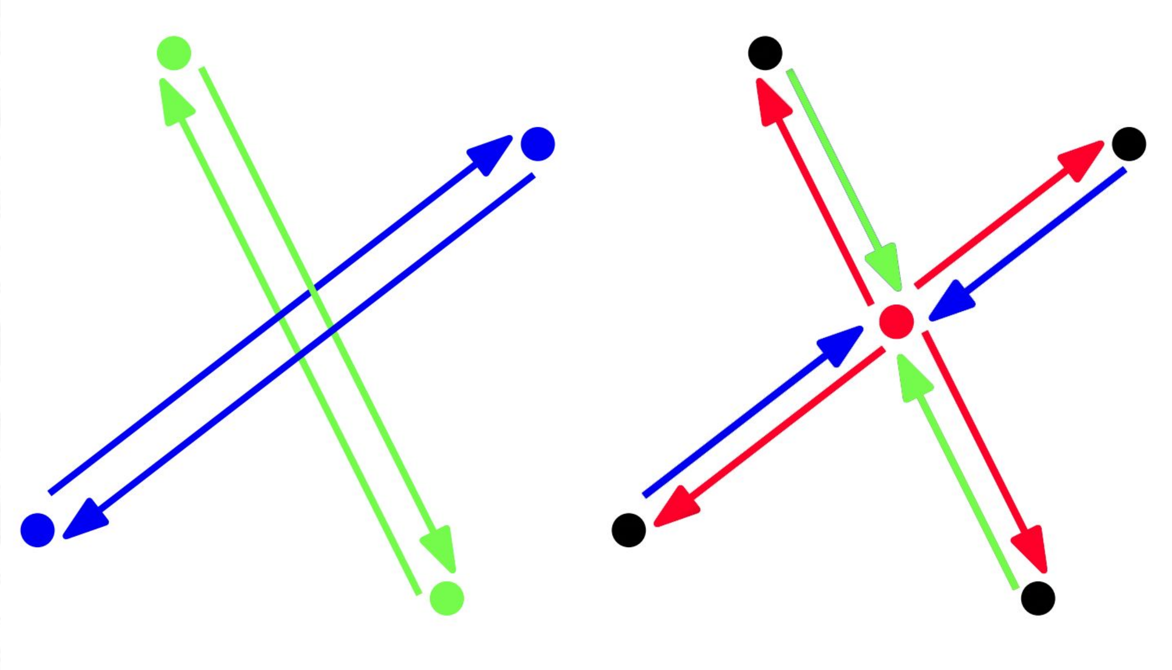
- *Existing half edges from S_1 (or S_2) will be edited* (origin point does not change, destination point changed to the intersection point).
- *New edges will be added* (origin at intersection, destination at the original edge's destination).



Frank Staals, http://www.cs.uu.nl/docs/vakken/ga/2021/

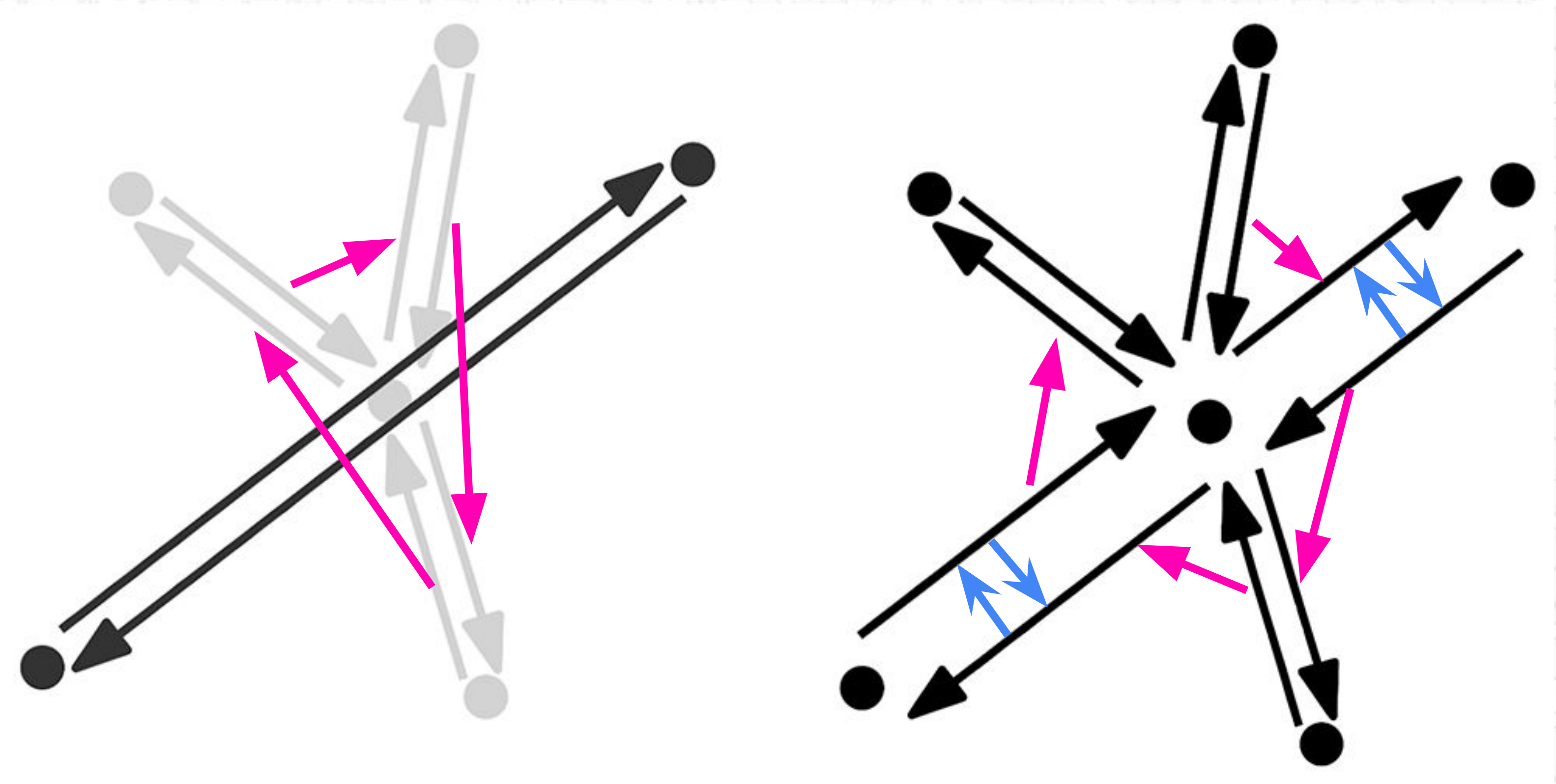
- *Existing half edges from S_1 (or S_2) will be edited* (origin point does not change, destination point changed to the intersection point).
- *New edges will be added* (origin at intersection, destination at the original edge's destination).

- New vertex will be added



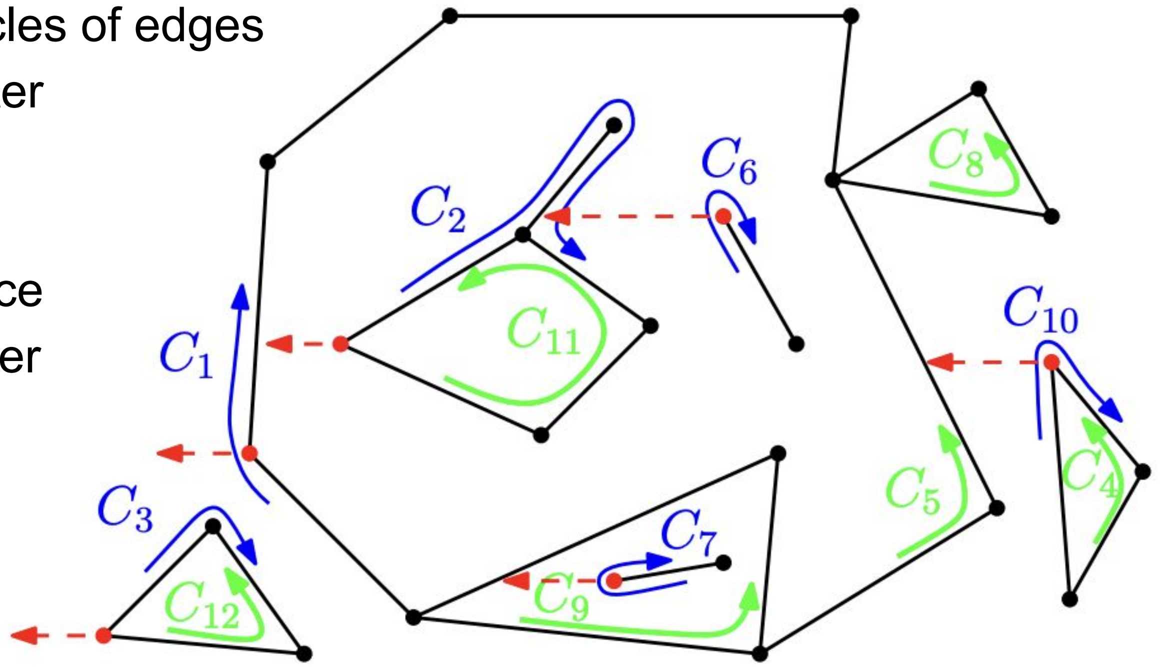
Frank Staals, http://www.cs.uu.nl/docs/vakken/ga/2021/

- Symmetric / opposite edges (re-)connected
- Next edge cycles updated



Construct Faces of the New Subdivision

- Determine cycles of edges
- Determine outer boundaries
- Create the unbounded face
- Determine inner components (if any) of each face
- Determine connected components



$C_1 \rightarrow$

$C_3 \rightarrow$

$C_2 \rightarrow C_5$

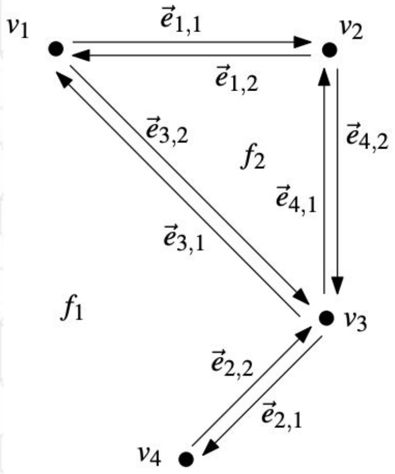
$C_7 \rightarrow C_9$

$C_6 \rightarrow C_2$

$C_{10} \rightarrow C_1$

Frank Staals, http://www.cs.uu.nl/docs/vakken/ga/2021/

Outer Component / Inner Component / Incident Face



Vertex	**Coordinates**	**IncidentEdge**
v_1	$(0,4)$	$\vec{e}_{1,1}$
v_2	$(2,4)$	$\vec{e}_{4,2}$
v_3	$(2,2)$	$\vec{e}_{2,1}$
v_4	$(1,1)$	$\vec{e}_{2,2}$

Face	**OuterComponent**	**InnerComponents**
f_1	**nil**	$\vec{e}_{1,1}$
f_2	$\vec{e}_{4,1}$	**nil**

Half-edge	**Origin**	**Twin**	**IncidentFace**	**Next**	**Prev**
$\vec{e}_{1,1}$	v_1	$\vec{e}_{1,2}$	f_1	$\vec{e}_{4,2}$	$\vec{e}_{3,1}$
$\vec{e}_{1,2}$	v_2	$\vec{e}_{1,1}$	f_2	$\vec{e}_{3,2}$	$\vec{e}_{4,1}$
$\vec{e}_{2,1}$	v_3	$\vec{e}_{2,2}$	f_1	$\vec{e}_{2,2}$	$\vec{e}_{4,2}$
$\vec{e}_{2,2}$	v_4	$\vec{e}_{2,1}$	f_1	$\vec{e}_{3,1}$	$\vec{e}_{2,1}$
$\vec{e}_{3,1}$	v_3	$\vec{e}_{3,2}$	f_1	$\vec{e}_{1,1}$	$\vec{e}_{2,2}$
$\vec{e}_{3,2}$	v_1	$\vec{e}_{3,1}$	f_2	$\vec{e}_{4,1}$	$\vec{e}_{1,2}$
$\vec{e}_{4,1}$	v_3	$\vec{e}_{4,2}$	f_2	$\vec{e}_{1,2}$	$\vec{e}_{3,2}$
$\vec{e}_{4,2}$	v_2	$\vec{e}_{4,1}$	f_1	$\vec{e}_{2,1}$	$\vec{e}_{1,1}$

** not covered in detail*

Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 2

Outline for Today

- Questions about Homework 1?
  Questions about CGAL/Qt installation?
- Today's Motivation
- Minimal Representation (e.g., Essentially Data File Formats)
- Proper Data Structures w/ Adjacency
- Line Sweep Algorithm for Map Overlay
  - Enumerate Intersection Cases for Map Overlay
  - Update Edges, Vertices, and Faces
  - Analysis
- Next Time

Analysis

- Let S_1 be a subdivision of complexity n_1,
  let S_2 be a subdivision of complexity n_2, and let $n = n_1 + n_2$.
- The overlay of S_1 and S_2 can be constructed in $O(n \log n + k \log n)$ time, where k is the complexity of the overlay.

  - Copying the edges from S_1 and S_2 takes $O(n)$ time
  - The planar sweep takes $O(n \log n + k \log n)$ time [prev. lecture]
  - Constructing the faces take $O(k)$ time.
  - Labeling the faces with the face attributes from S_1 and S_2 is $O(n \log n + k \log n)$ * *not covered in detail*

Analysis

Complexity is # of edges or # of vertices + # of faces

$V + F = E + 2$

- S_1 has complexity n_1
- S_2 has complexity n_2
- $n = n_1 + n_2$

- k is the complexity of the overlay of S_1 and S_2

- In the worst case:

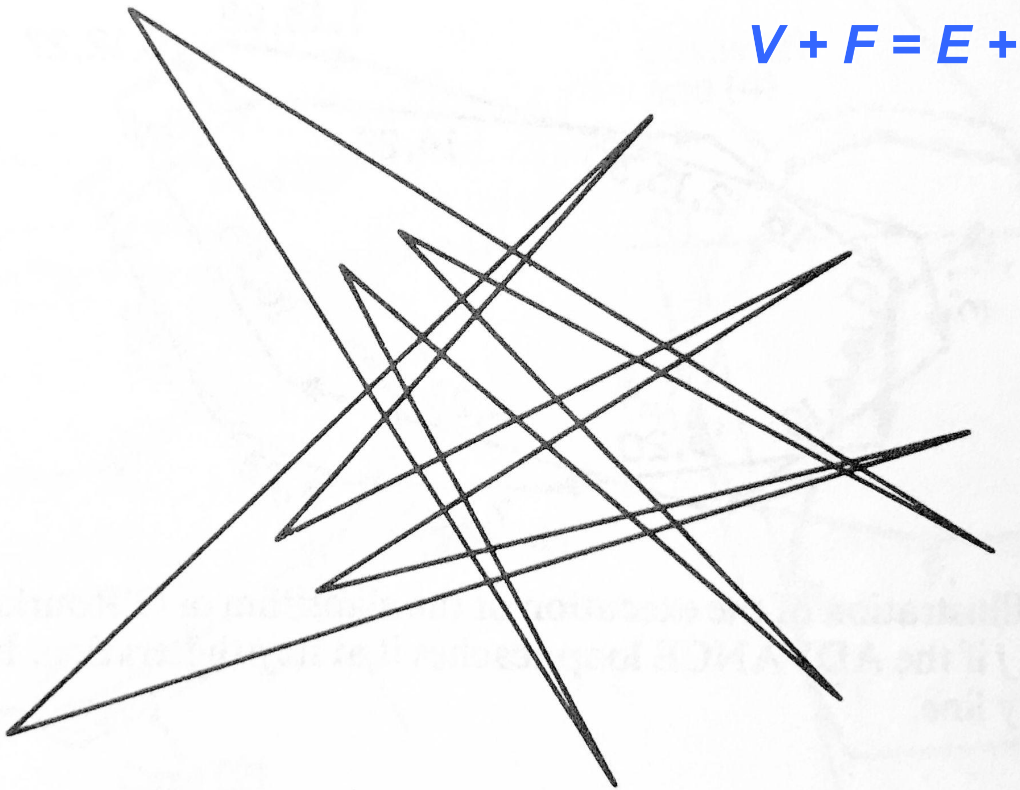


Figure 7.11 The intersection of two star-shaped polygons.

Computational Geometry: An Introduction
Preparata & Shamos, Springer 1985

Analysis

Complexity is # of edges or # of vertices + # of faces

V + F = E + 2

- S_1 has complexity n_1
- S_2 has complexity n_2
- $n = n_1 + n_2$

- k is the complexity of the overlay of S_1 and S_2

- In the worst case:
  k is $O(n_1 * n_2) = O(n^2)$

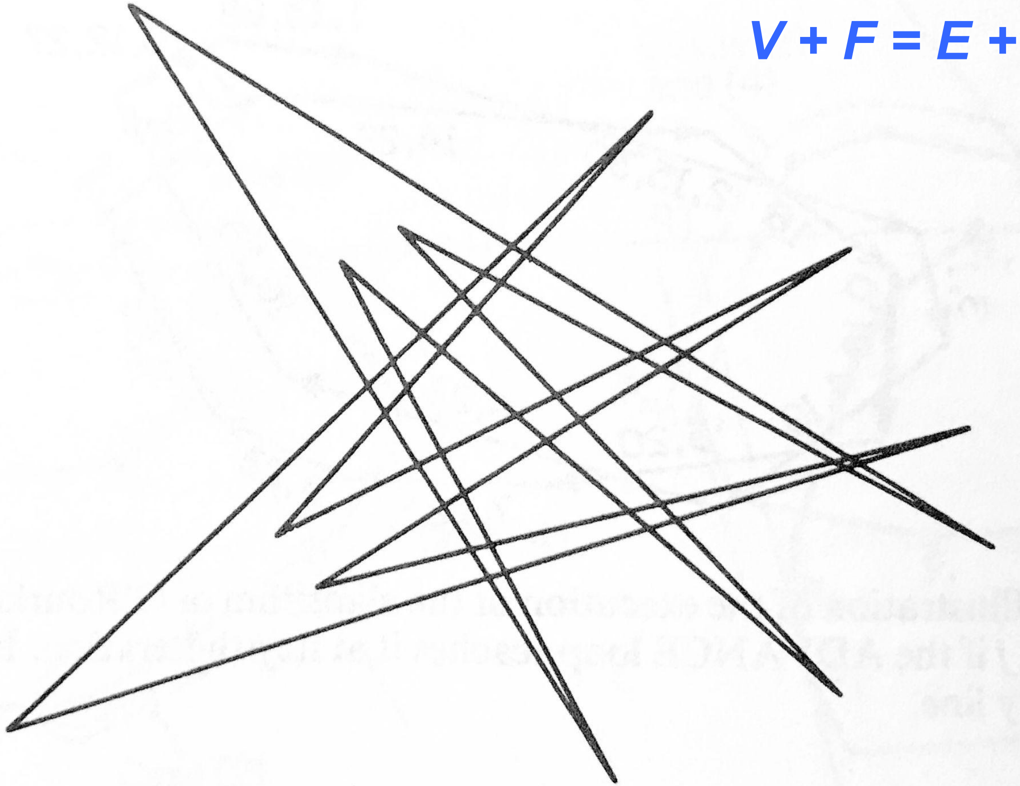


Figure 7.11 The intersection of two star-shaped polygons.

Computational Geometry: An Introduction
Preparata & Shamos, Springer 1985

Outline for Today

- Questions about Homework 1?
  Questions about CGAL/Qt installation?
- Today’s Motivation
- Minimal Representation (e.g., Essentially Data File Formats)
- Proper Data Structures w/ Adjacency
- Line Sweep Algorithm for Map Overlay
- Next Time

Next Time… Polygon Triangulation