CAD & Computational Geometry

Course plan

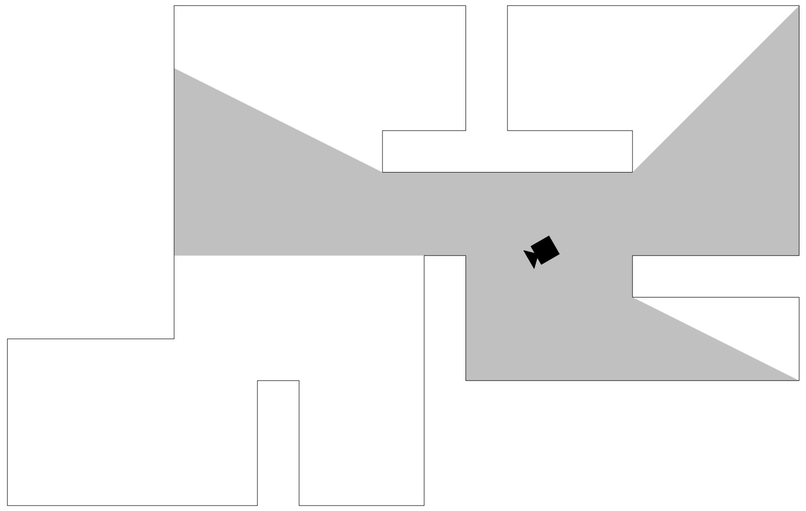
- Introduction
- Segment-Segment intersections
- Polygon Triangulation
- Delaunay Triangulations
- Geometric Search
- Voronoï Diagrams

Polygon Triangulation

- Applications
  - Graphical display of polygons
    - Graphic cards in computers know how to display line segments and triangular patches, and points.
    - Every complex entity must be decomposed into triangles or lines, and displayed as such. It is the case of polygonal patches
    - This decomposition is made by the graphic card (using the GPU) or by the host computer (in the graphic driver), even if the user believes that the software interface allows to display polygons in a native way.
    - This is in fact the case for the visualization code used in this course...

Polygon Triangulation

- Another application
  - “Art gallery” Problem
    - Be able to supervise a complete floor in an art gallery using an adequate (minimal) number of video cameras placed in appropriate locations

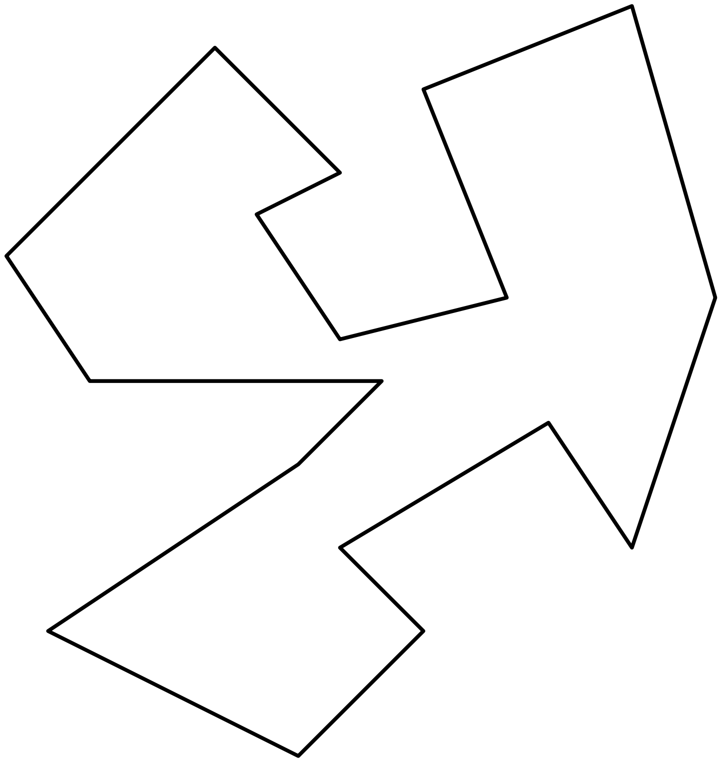


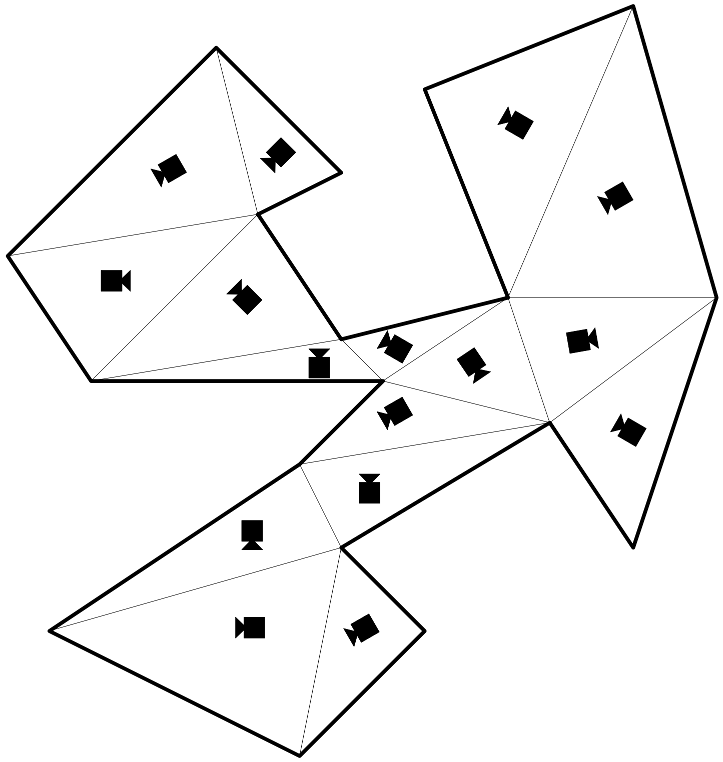
Polygon Triangulation

- Link between the art gallery problem and the triangulation of a polygon P
  - Ex. Convex Polygon : may be supervised with only one camera
  - What is the number of cameras needed to supervise an n-sided polygon
    - We do not look for the smallest number ! (hard problem – NP complete)
    - The answer is not trivial : an n-sided polygon may have a very complex shape.
    - We'll try to decompose P into triangles, which are easy to supervise individually, since triangles are necessarily convex.

Polygon Triangulation

- We'll transform the shape into triangles by linking non adjacent vertices on the contour.

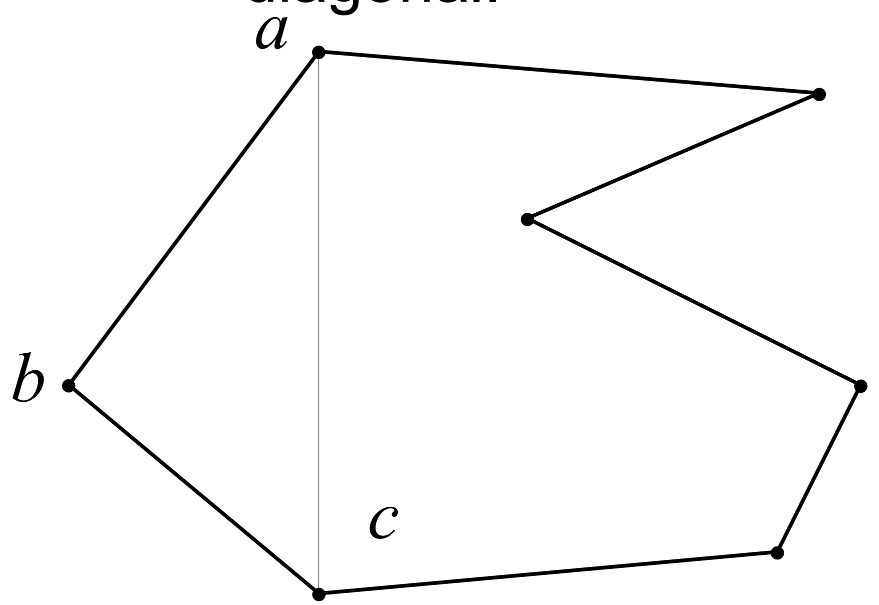




Polygon Triangulation

- Does such a triangulation exist for an n-sided polygon : proof by induction.

  For n=3, it is trivial. Let $n > 3$ and let us state that the triangulation exists for all $m < n$. We will show that a diagonal slicing the polygon exists. Let b the leftmost vertex, and a and c the two adjacent vertices to b ; If $\overline{ac}$ is completely inside the polygon, then we have found a diagonal.



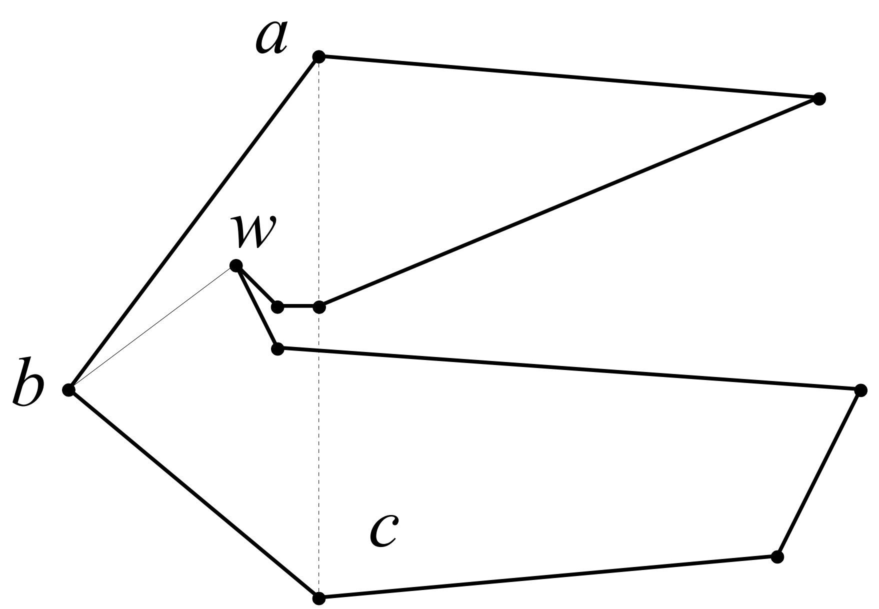
The polygon may be decomposed in two : one triangle with $p=3$ vertices and another polygon with $q < n$ vertices, with $p+q=n+2$. (in our case, $q=n-1$)

Polygon Triangulation

If the segment $\overline{ac}$ is not entirely contained inside, then let us consider the vertices located inside the triangle abc.

Among these, let w be the furthest away from $\overline{ac}$.

Then, $\overline{bw}$ is a diagonal. (If it is not be the case -$\overline{bw}$ intersects some of the sides of the polygon- then it means that w was not the point furthest away from $\overline{ac}$, which is contradictory.)

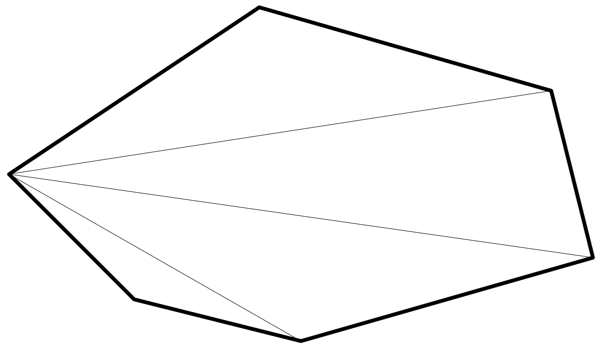


In this case, the polygon is cut in two, one with p vertices, the other with q vertices, with $p < n$, $q < n$ and $p+q=n+2$.

- In both cases, the slicing may be repeated until only triangles remain.

Polygon Triangulation

- How many triangles ?



  - For a convex polygon, it is easy: $m=n-2$ triangles. Let's suppose this true for any polygon, in particular for polygons having $p,q < n$ vertices.

    By taking a diagonal to splice the polygon in two,as in the steps before, one gets $p+q=n+2$, with $p < n$, and $q < n$.

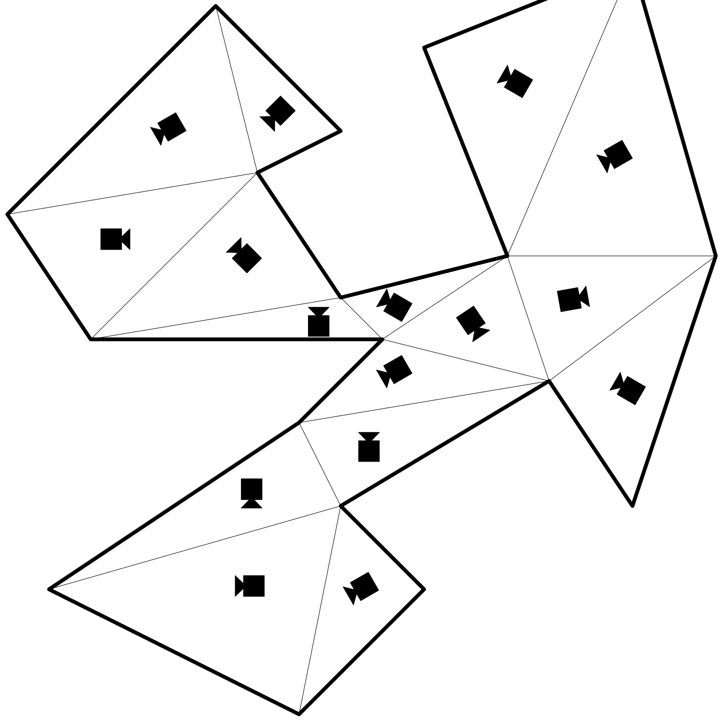
    Each sub-polygon is decomposed into $m_p=p-2$ and $m_q=q-2$.

    Therefore, the initial polygon is decomposed into $m = m_p+m_q = p-2+\text{q}-2 = n+2-4 = n-2$ triangles. QED

Polygon Triangulation

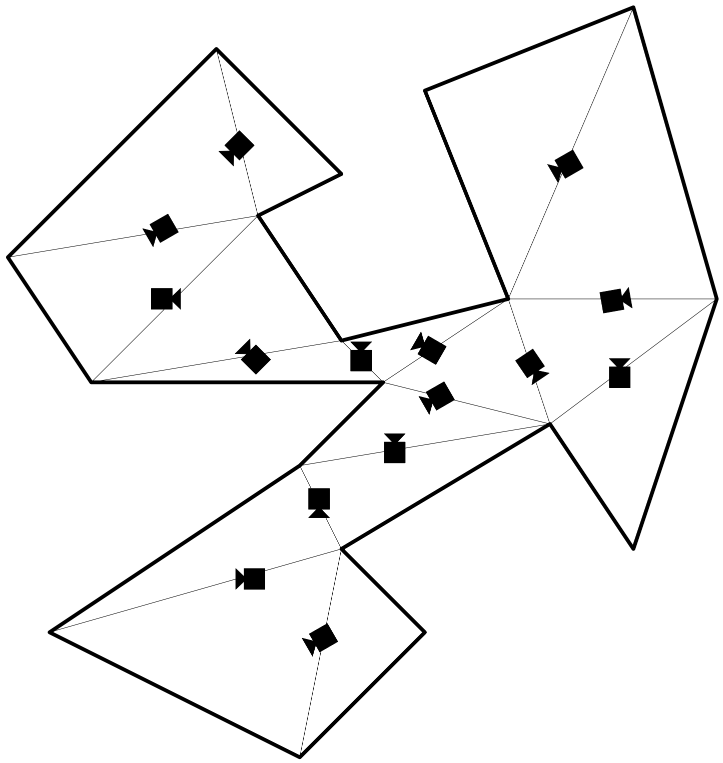
- Let’s go back to the initial problem

1 camera per triangle



$n-2$ cameras

1 camera per edge

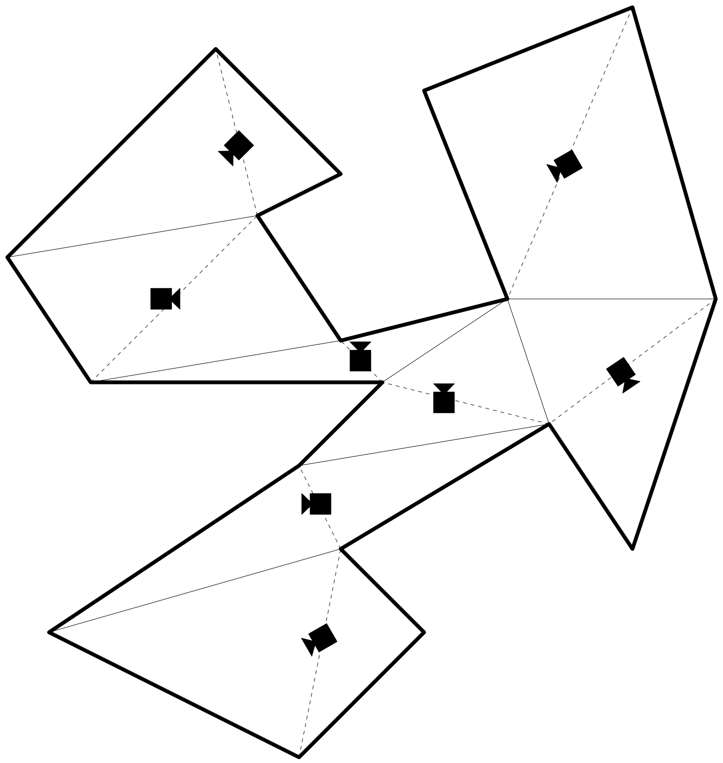


$n-3$ cameras

Polygon Triangulation

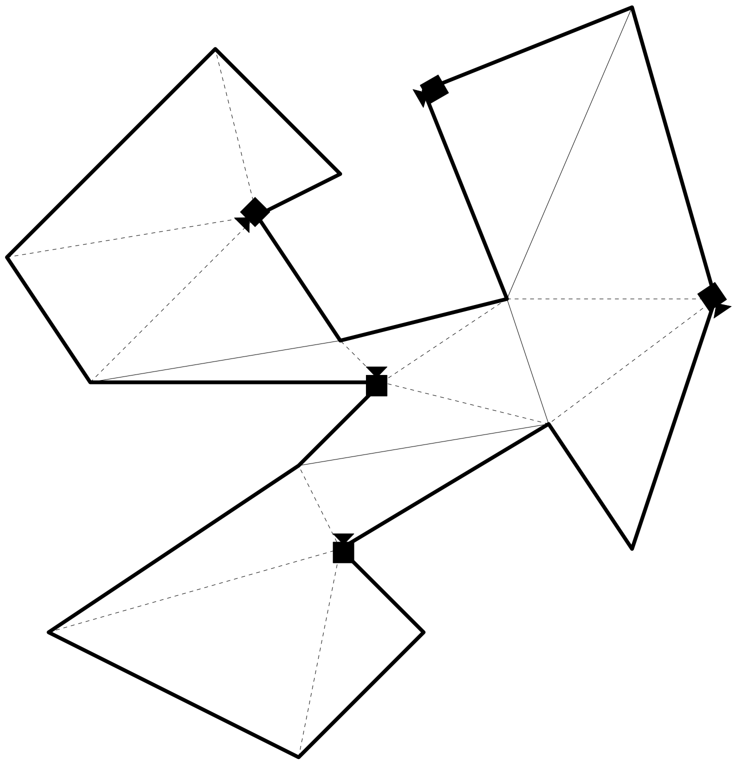
- Let’s go back to the initial problem

1 camera per pair of triangles



$\sim n/2$ caméras

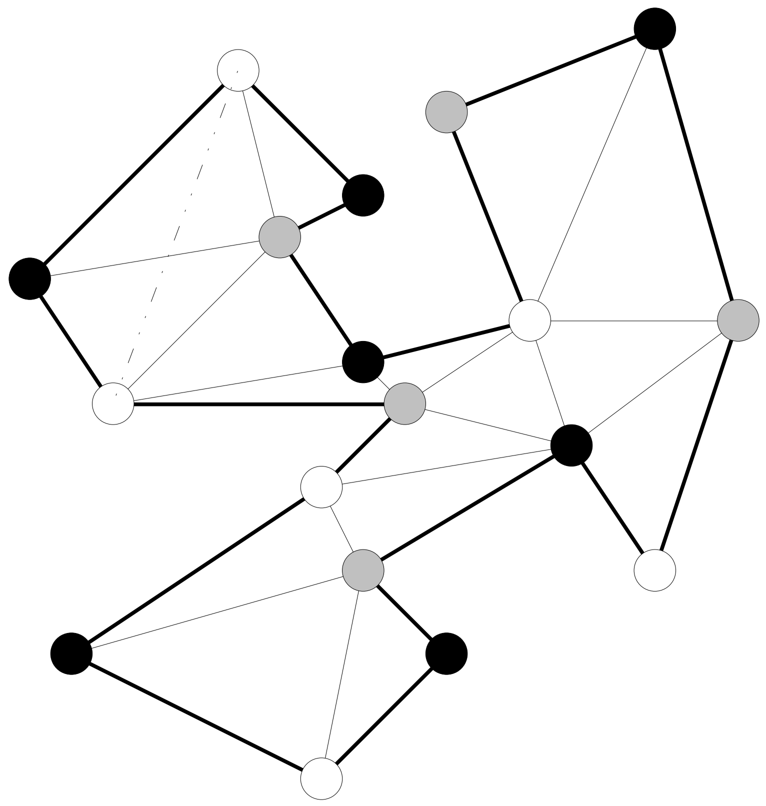
1 camera for some vertices



$\sim n/3$ caméras ?

Polygon Triangulation

- How to determine the sites for cameras ? → 3-coloring

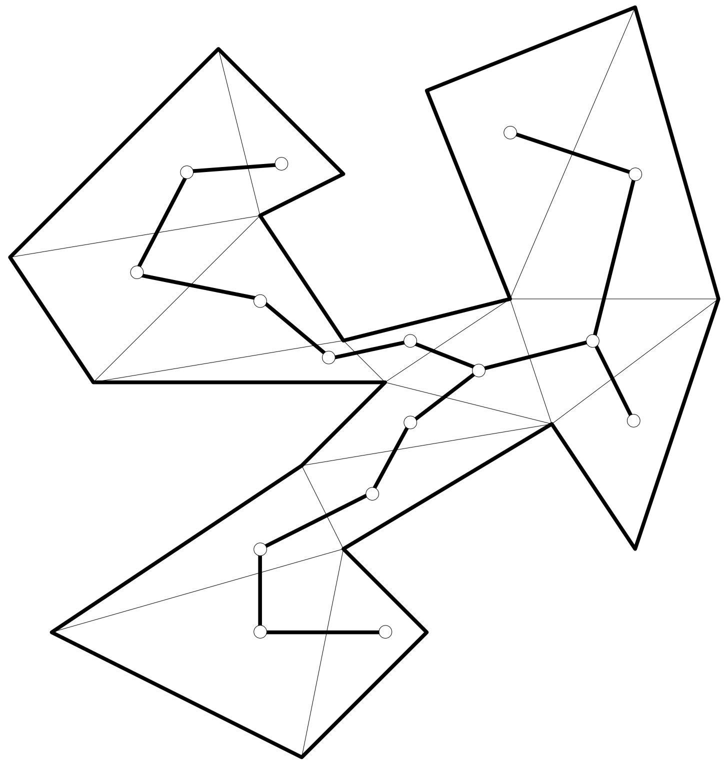


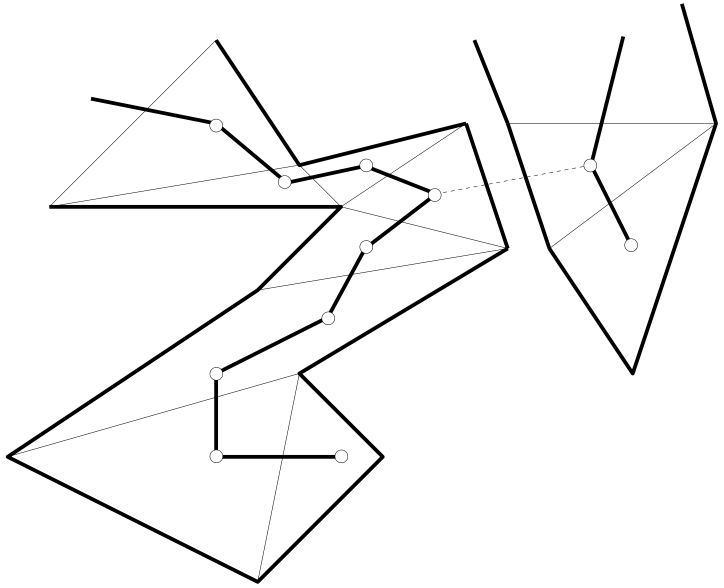
- Each edge must link to two distinct colors
- Cameras will be located on vertices of a give color (e.g. grey)
- The 3-coloring *depends* on the triangulation (if it is unique, it is only for a given triangulation)
- Does such a coloring always exist ?

CAD & Computational Geometry

Polygon Triangulation

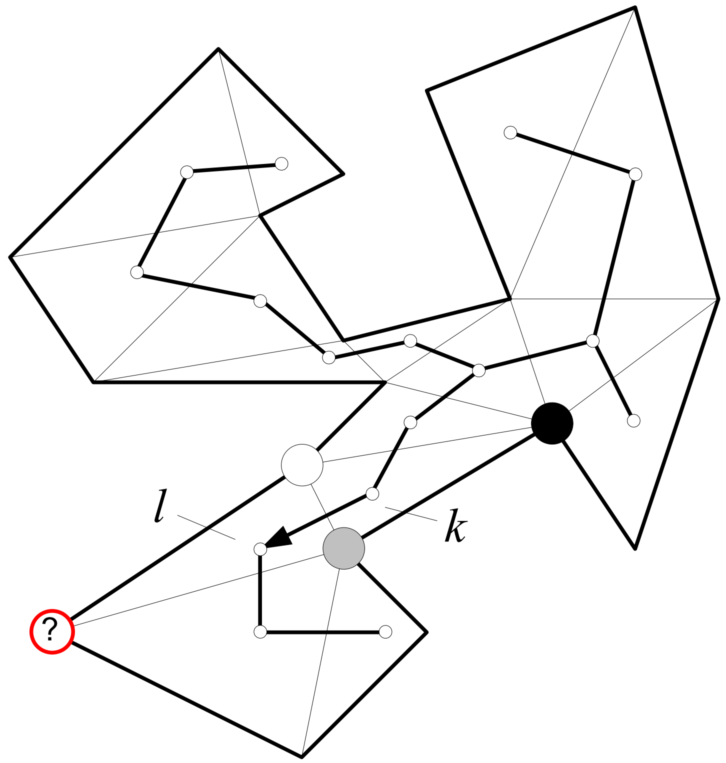
- Existence of the 3-coloring.
  - Lets consider the connectivity graph of the triangles (dual graph of the triangulation T)
  - It is obvious that it is a tree, because if one takes a diagonal out of T, then the graph becomes disjoint because we cut the triangulation into two.





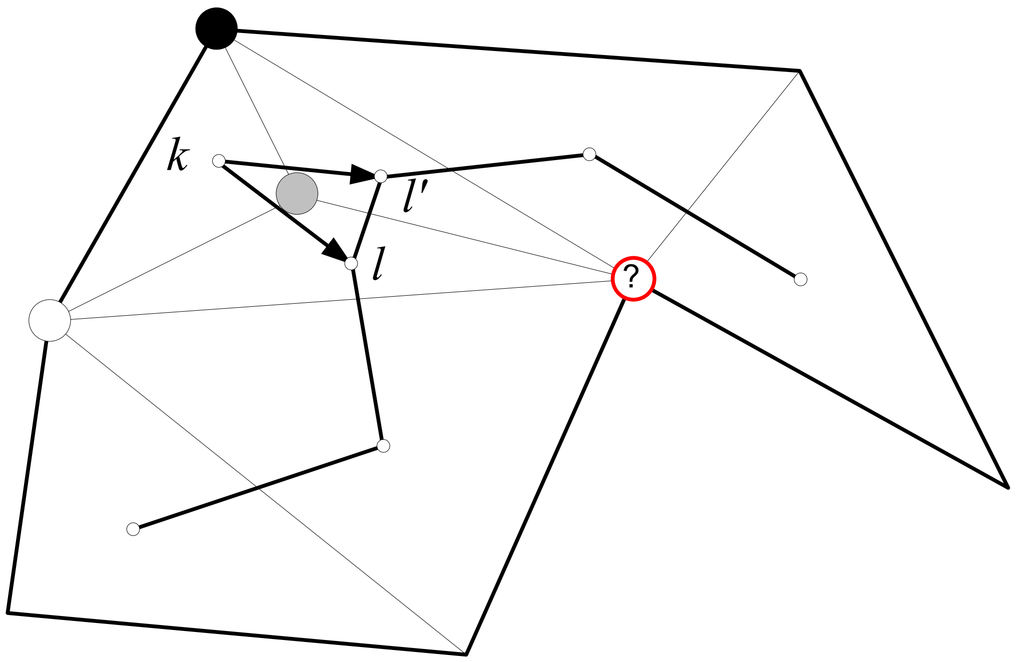
Polygon Triangulation

- Existence of the 3-coloring.
  - One can walk along the tree from any vertex k. At each new vertex, l for instance , one knows the both triangles $t(k)$ and $t(l)$ share an edge, and that edge is already colored. Only one color choice for the remaining vertex of $t(l)$.
  - Every time on jumps from triangles to triangle in the graph, one uses one of the remaining edges of the tree T ; therefore the choice of the color for the next vertex is always possible.
  - The coloring is done in $O(n)$ operations.



Polygon Triangulation

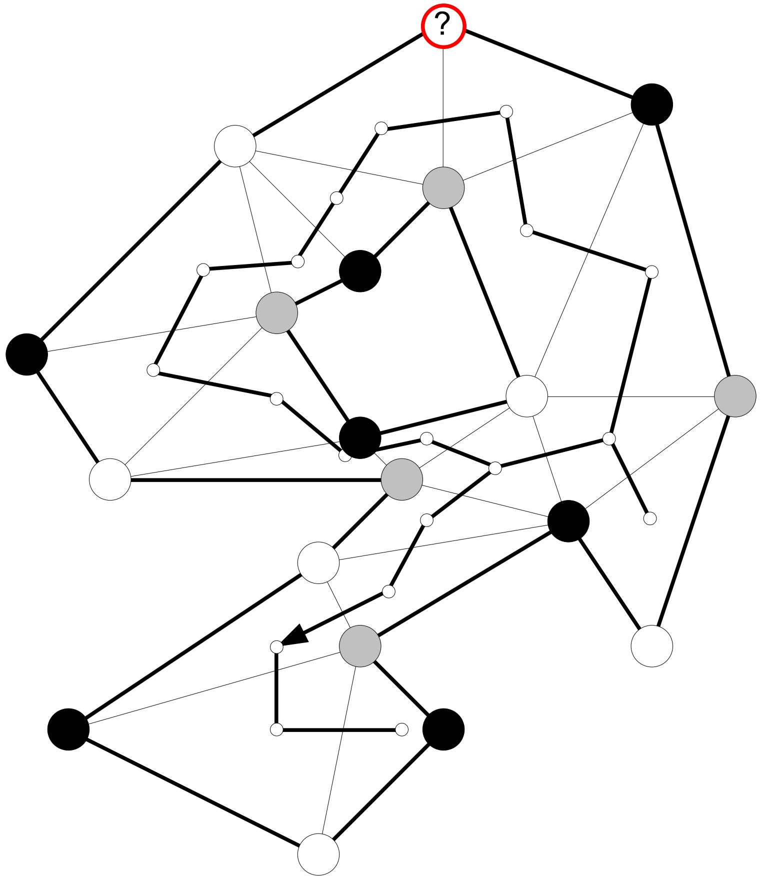
- Note :
  - The 3-coloring works only if the graph is a tree.
  - Otherwise, it is easy to find a counterexample : cyclic graph because of an internal vertex



A 4th color is at least needed

Polygon Triangulation

- Cycles coming from holes in the polygon

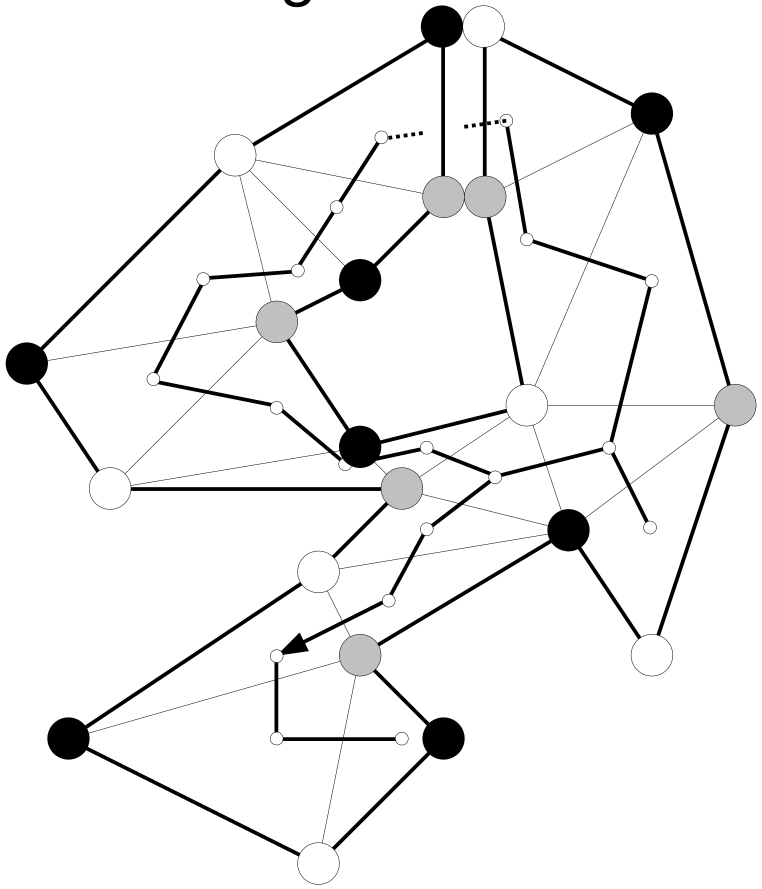


- Solution : Change the patch into a simple polygon by doubling some vertices

CAD & Computational Geometry

Polygon Triangulation

- Solution : Change the patch into a simple polygon by doubling some vertices



Polygon Triangulation

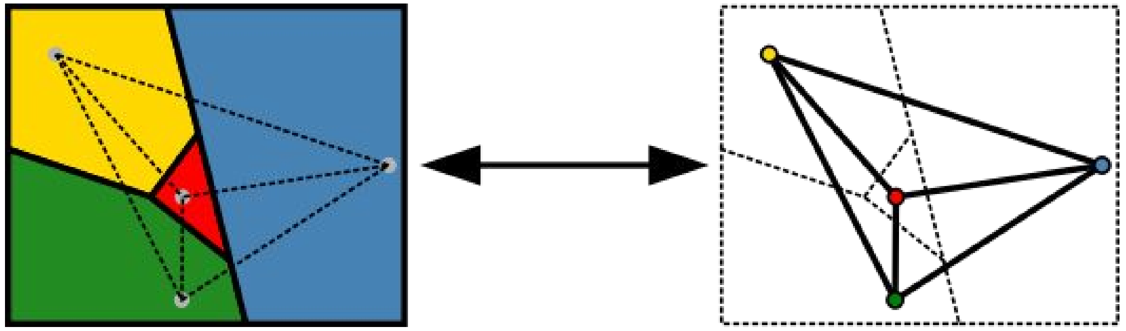
- In fact, the 4-coloring always exists for a planar graph (hence for a triangulation in the plane)

  The conjecture has been proposed in 1852 by a british botanist, who found that coloring the counties of England needed only 4 colors.

  - Proof only in 1977 using... computers.

    Appel, K. and Haken, W. “The Solution of the Four-Color Map Problem” Scientific American 237, 108-121, 1977.

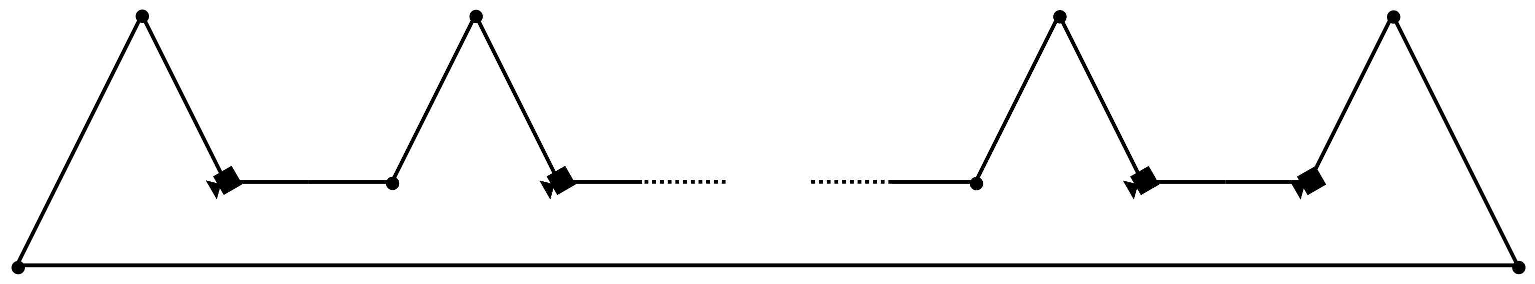
    Appel, K. and Haken, W. “Every Planar Map is Four Colorable”, Contemporary Mathematics 98, Amer. Math. Soc., 1989



Polygon Triangulation

- Because of the 3-coloring, the number of cameras is the integer part of $n/3$, in the worst case.

  It is often possible to use less cameras (e.g. for a convex polygon,1 is enough), however there are polygons for which the figure above is a minimum.



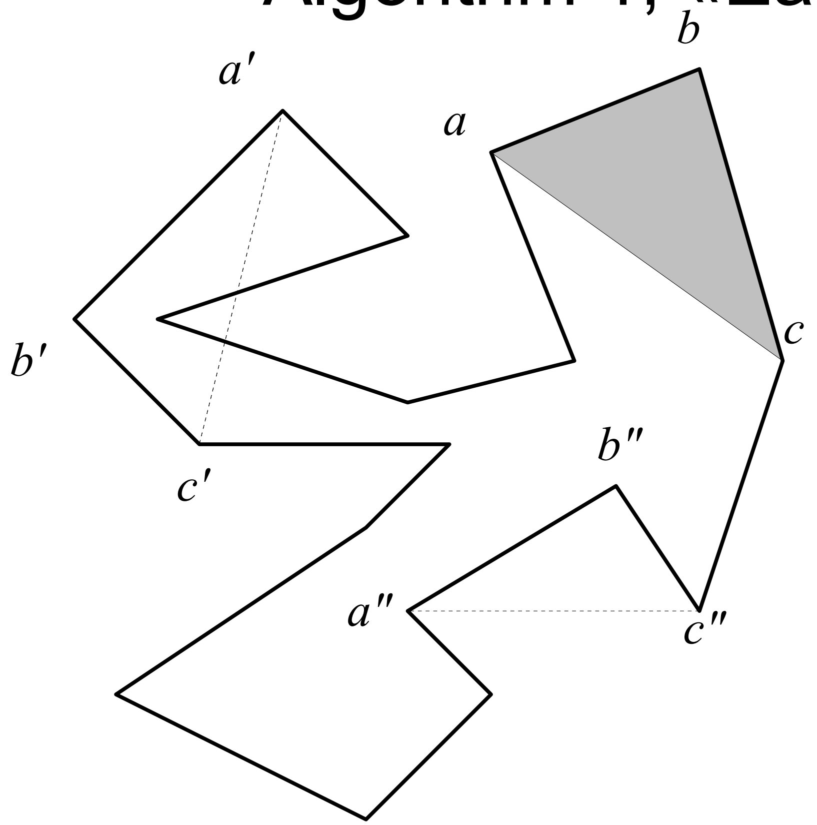
Exactly $n/3$ teeth, exactly $n/3$ cameras.

Polygon Triangulation

- Back to our problem...

  We need a triangulation

  - Algorithm 1, «Ear clipping»



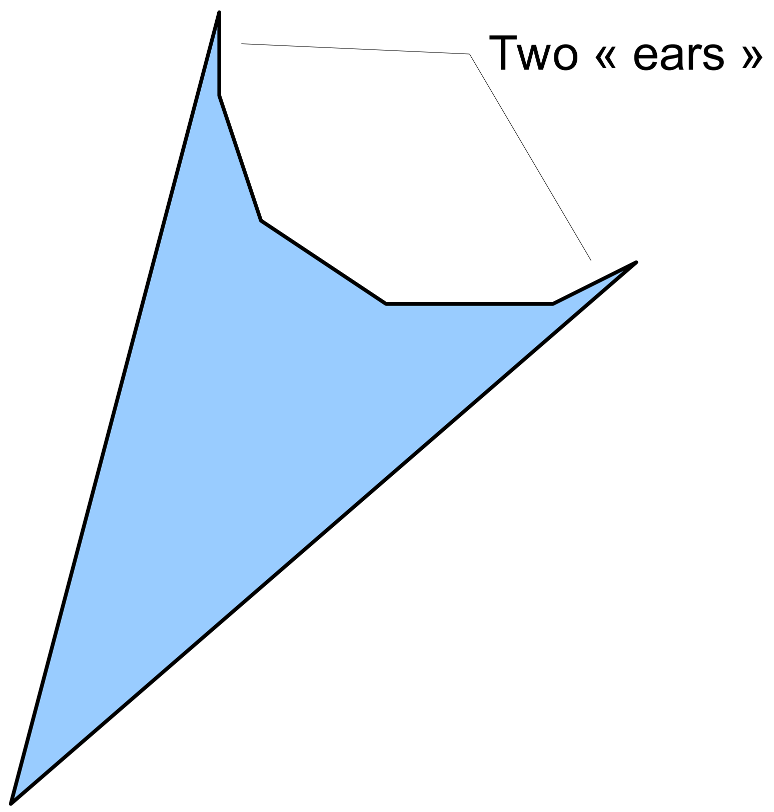
An « ear » is a sequence a, b, c of consecutive vertices forming a triangle entirely in the polygon, which does not contain other vertices.

- The principle is to withdraw such ears as they appear ...

  Does a polygon always have ears ? - YES ! - it even always have two ears : cf

  Meisters, G. H., "Polygons have ears." American Mathematical Monthly 82 (1975). 648-651

Polygon Triangulation



Polygon Triangulation

Algorithm 1, « Ear clipping ».

- A naïve implementation leads to a $O(n^3)$ complexity

  Find an ear – $O(n^2)$ ◄

  Eliminate it – $O(1)$

  Find a new ear – $O((n-i)^2)$

  Repeat n–2 times

- It is possible to do better : in $O(n^2)$

  Arrange the vertices in 4 lists :
  - one for every vertex (circular list) – $O(n)$
  - one contains the reflex vertices (concave) – $O(n)$
  - one containes the convex vertices – $O(n)$
  - one contains all the ears – $O(n^2)$

  Take the 1st ear; eliminate it – $O(1)$ ◄

  Check for neighboring vertices in the lists:
  - If it was a convex vertex, it remains so
  - If if was an ear, it remains so
  - If it was a reflex vertiex, it may change to be convex, or an ear

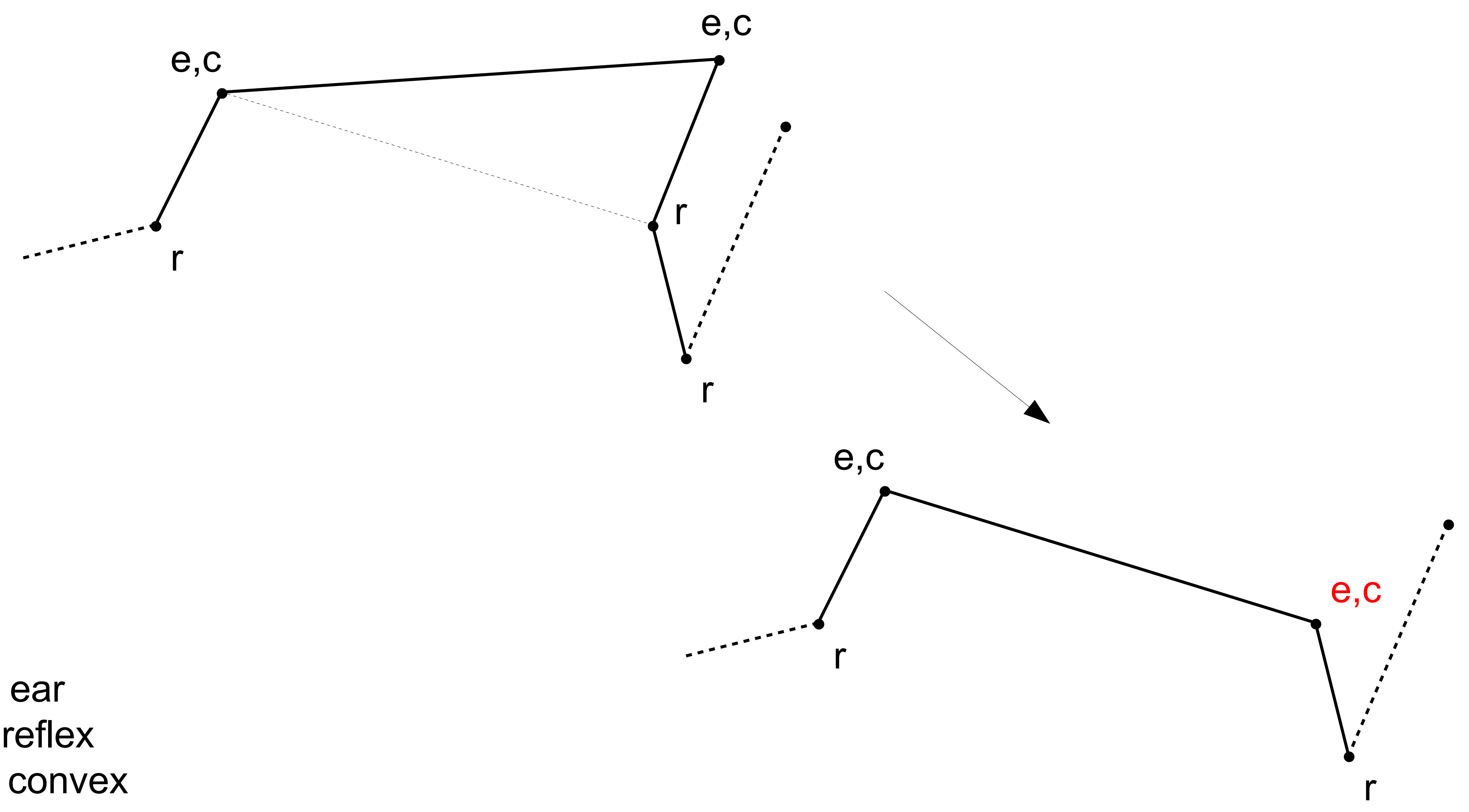
  On needs therefor test if it becomes a new ear – in $O(n-i)$

  Update all lists – $O(1)$

  n–2 times

CAD & Computational Geometry

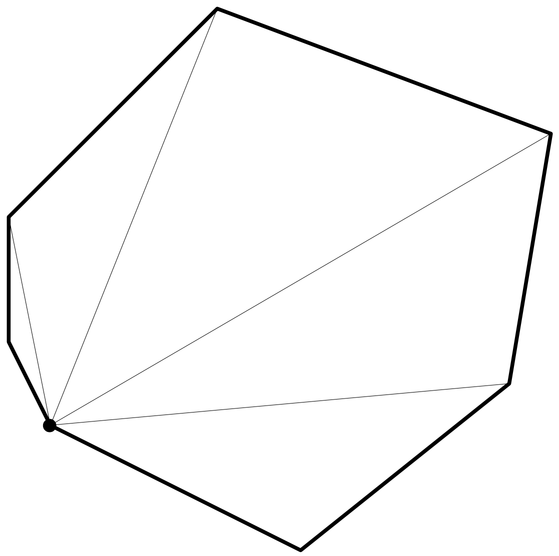
Polygon Triangulation



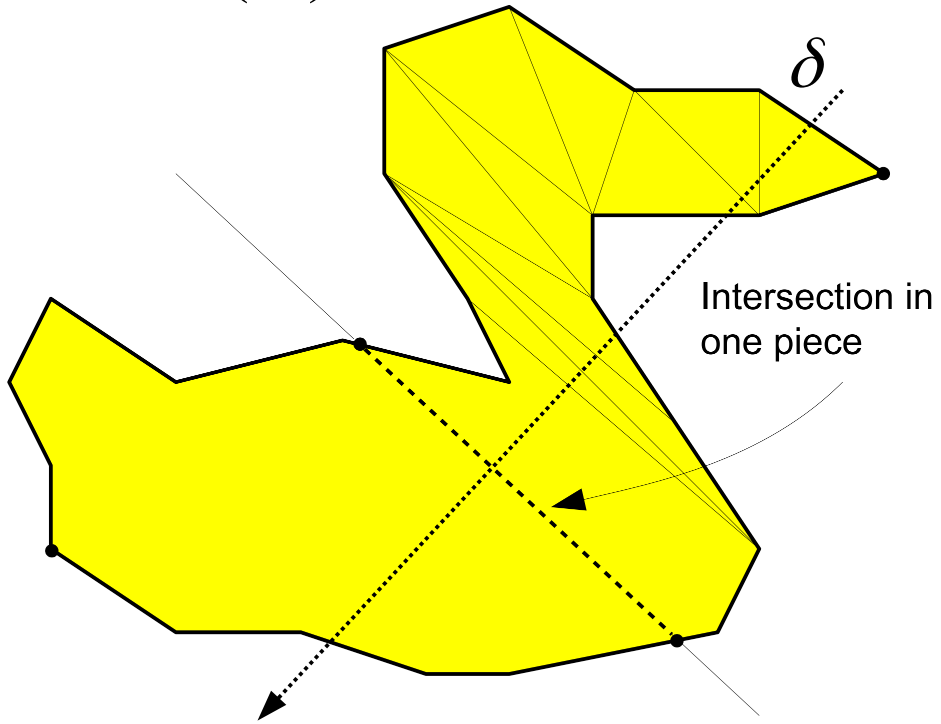
e = ear
r = reflex
c = convex

Polygon Triangulation

- Is it possible to do better than $O(n^2)$?



Convex polygon : $O(n)$



Monotonous polygon
with respect to an orientation δ : $O(n)$
(formal demonstration later)

Polygon Triangulation

- Second algorithm, use of a decomposition into simpler polygons

  Idea : decompose in sub-polygons that have a simpler shape so that the triangulation is deemed more simple. However : decomposition into convex polygons as difficult as the triangulation itself – remains monotone polygons

  Algorithm 2 has therefore two parts :

  – Decomposition into monotone sub-polygons (we hope in $o(n^2)$, e.g. $O(n\log n)$)

  – Triangulation of the sub-polygons

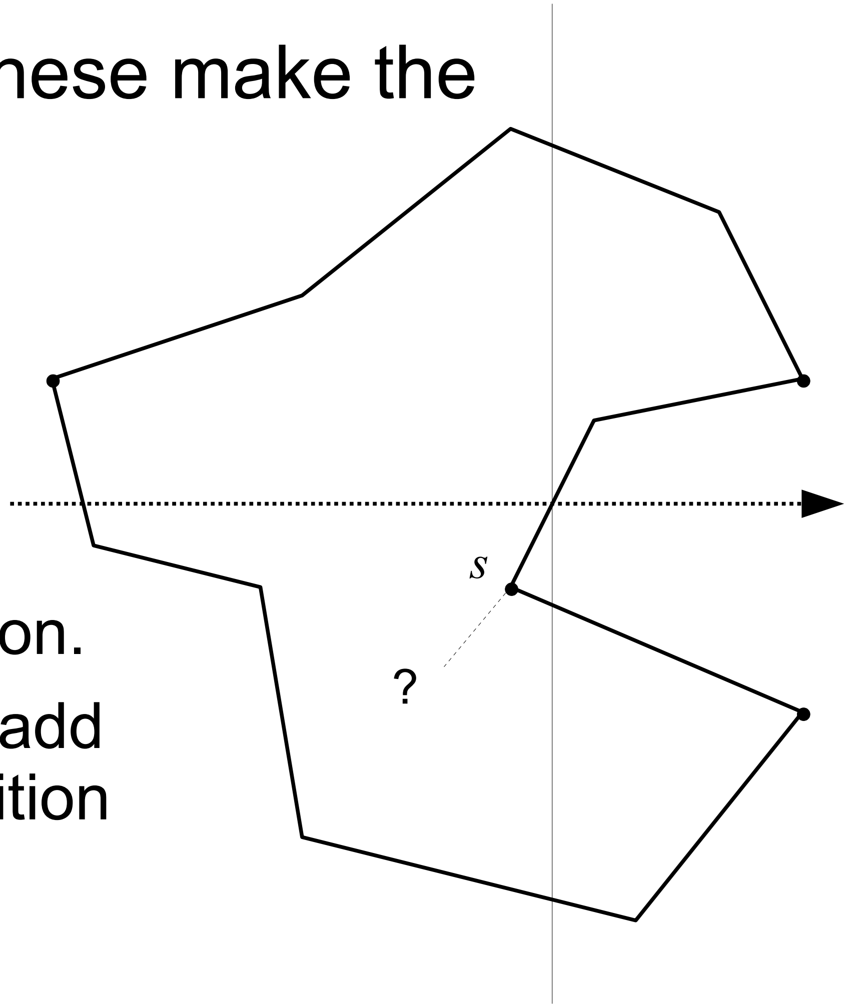
CAD & Computational Geometry

Polygon Triangulation

- Decomposition along x

  Some special vertices do exist. These make the polygon non monotone

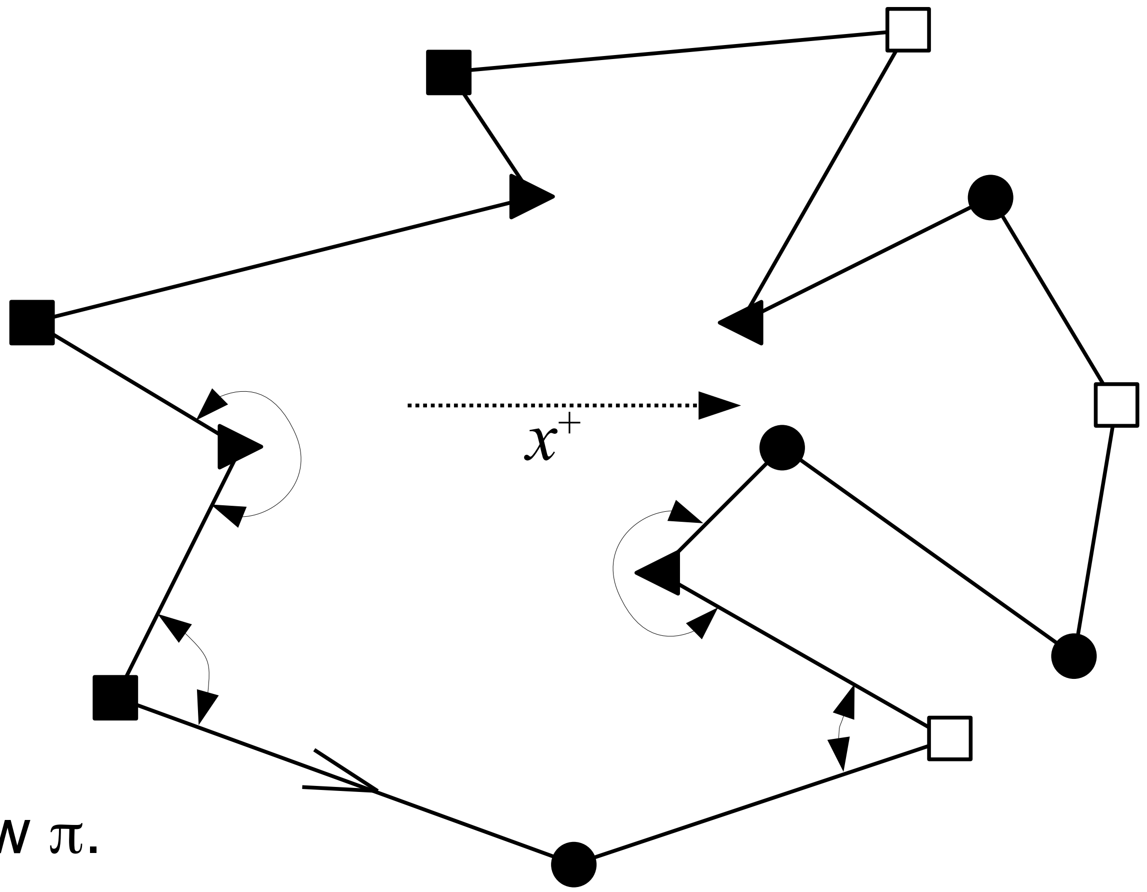
  - These are the vertices for which there is a change of the apparent orientation (with respect to the reference line) when one follows the exterior of the polygon.
  - It is from these vertices that we will add diagonals to achieve the decomposition



CAD & Computational Geometry

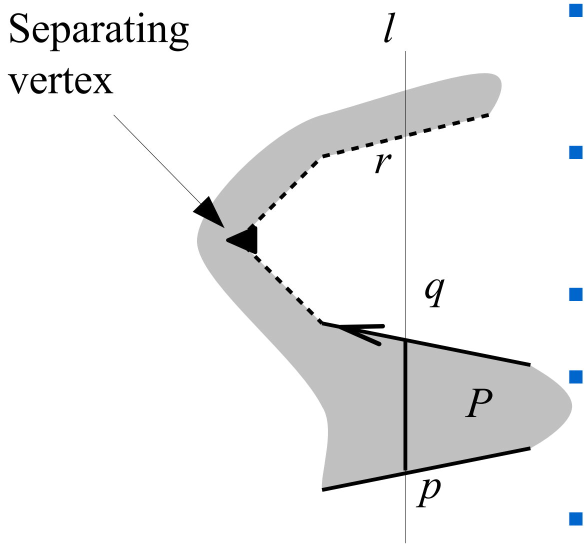
Polygon Triangulation

- Classification of vertices
  - ● « regular » vertex
  - ■ « start » vertex
  - □ « end » vertex
  - ◀ « separating » vertex
  - ▶ « fuse » vertex
  - The « **start** » vertex has two neighbors to the *right* and an interior angle below π.
    - If the angle is above π the vertex is « **separating** »
  - The « **end** » vertex has two neighbors to the left, and an interior angle below π.
    - If the angle is above π the vertex is « **fuse** »
  - In all other cases, the vertex is « **regular** »



Polygon Triangulation

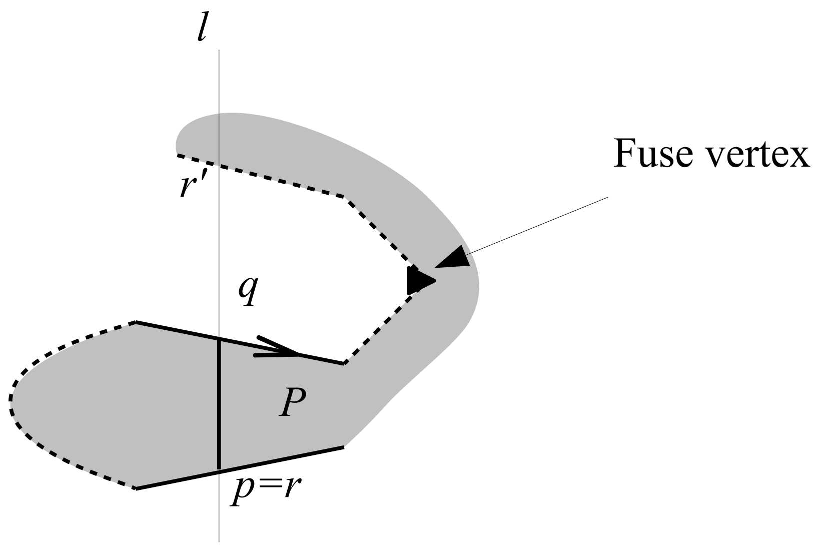
- It is clear that a monotone polygon does not contain any « fuse » or « separating » vertices
  - Same as to prove that a non monotone polygon contains at least one « fuse » or « separating » vertex.
  - Non x-monotone → a vertical line l intersects P in more than one connected component. Choose l such that the lowest intersection is a segment – but not a single vertex.



- Let p be the point under the boundary, and q the point above it.
- Lets start from q and go along the boundary so that the inside of the polygon is at the left of the boundary.
- At some point, l is intersected again, at point r.
- If $r \neq p$, then we fond an extremal vertex that is a separating vertex.
- If $r = p$, lets walk the other way round.

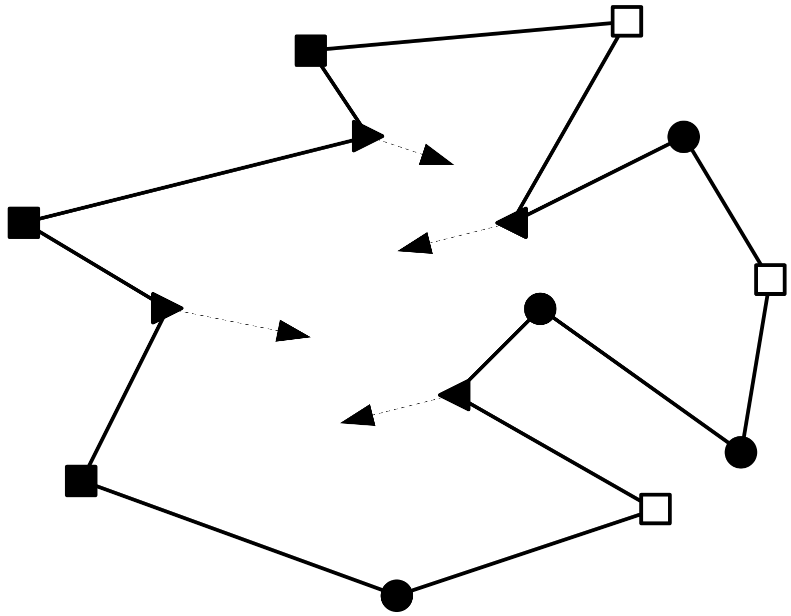
Polygon Triangulation

- If $r = p$, lets walk the other way round.. Like before, one intersects l at a new point r'. Necessarily, $r' \neq p$, otherwise it would mean that the intersection of l with P has only one connected segment.
- Thus, there exists one vertex to the right, which is necessarily a fuse vertex.
- Therefore, any non x-monotone polygon has at least one separating vertex and/or one fuse vertex.



Polygon Triangulation

- That means if one eliminates all separating and fuse vertices (*i.e.* transform them into other types of vertices), the original polygon would have been decomposed into monotone polygons.
  - It is done by adding diagonals : one to the left for each separating vertex, one to the right for each fuse vertex.
  - The difficulty lies in linking these diagonals to other vertices of the polygon.



Polygon Triangulation

- Use of the planar sweeping (again!)
  - Events are vertices of the original polygon (no new events are created)
  - Those are sorted in lexicographic order, in a priority queue.
  - The status T allows to build diagonals as the line sweeps to the right

Polygon Triangulation

- What to do when l meets an event ?
  - Case of « separating » vertices
    - One should link it to a close vertex so that we do not intersect other edges.

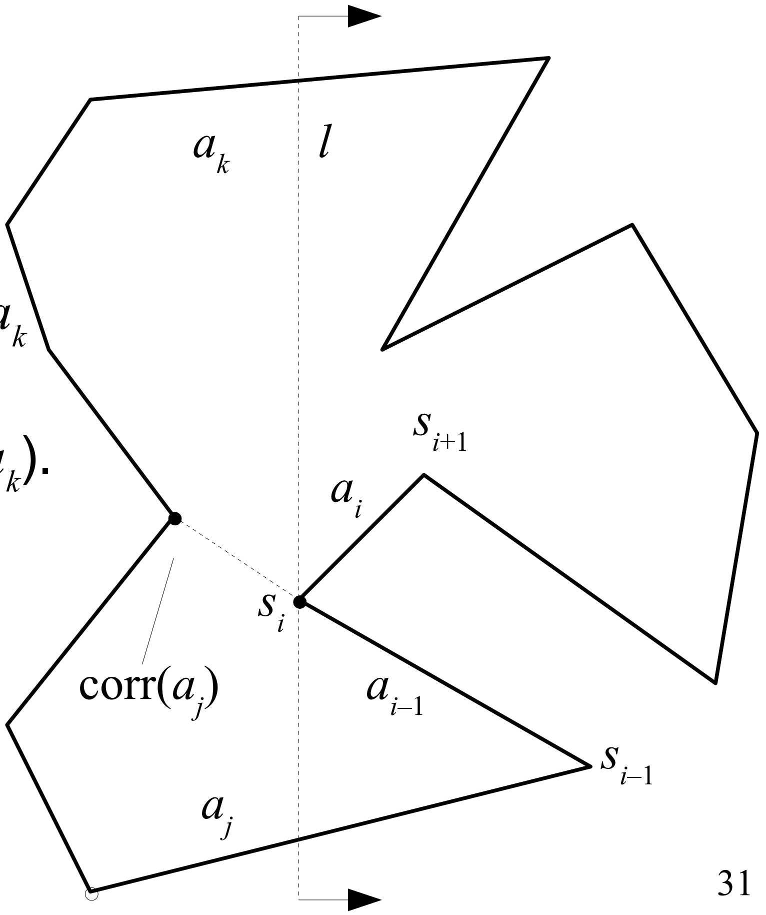
      Let a_j be the edge immediately below s_i along l, and a_k that immediately above. Then it is always possible to connect s_i to the rightmost vertex between a_j and a_k , and to the left of s_i. If it does not exist, simply use the leftmost vertex of a_j (or a_k).

      In every case, this vertex is marked as corresponding to a_j : $\mathrm{corr}(a_j)$.

s_i in a counter-clowise order

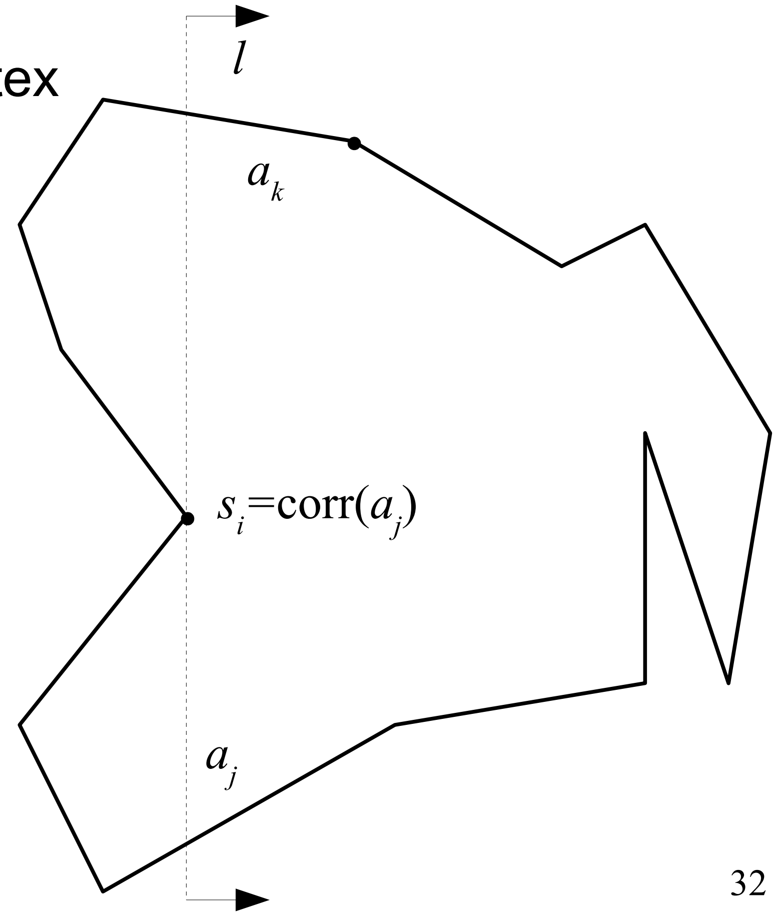
$a_i = \overline{s_i s_{i+1}}$

$a_{n-1} = \overline{s_{n-1} s_0}$



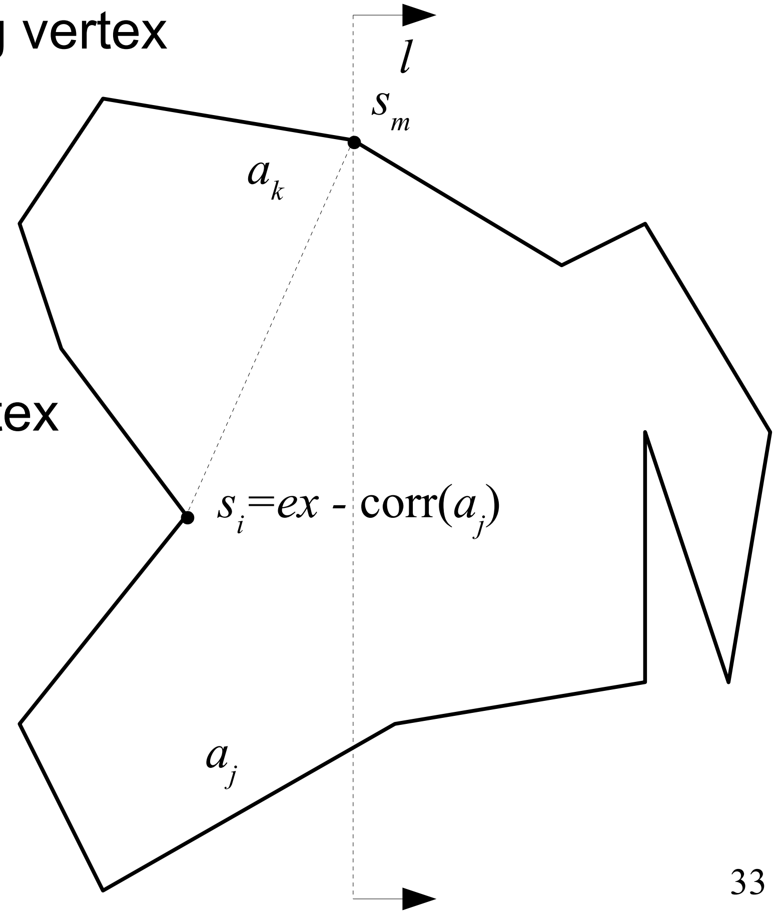
Polygon Triangulation

- What to do when l meets an event ?
  - Case of « Fuse » vertices
    - They must be linked to close vertices so that we do not cross edges – BUT it cannot be done immediately (the potential vertex being on the right of l)
    - One can notice that s_i can be marked as corresponding to de a_j at this point.



Polygon Triangulation

- What to do when l meets an event ?
  - Case of « Fuse » vertices
    - When l moves further, it meets s_m which becomes the new corresponding vertex of a_j .
    - It is a this point that one can link s_m with s_i by a diagonal.
    - This is done only if s_i is a fuse vertex !
    - It is possible that s_m is a separating vertex it is good - one diagonal for two.



Polygon Triangulation

- Status to be updated at each event
  - Binary search tree T – contains the edges for which the interior of the polygon is “above”.
  - The ordering in this tree is based on the vertical position : lower edges are found before upper edges.
  - For each edge, the vertex to which it “corresponds” is stored
- For every event s_i , the status is updated :
  - s_i is a « start » event : insert edge a_i in T, update $\mathrm{corr}(a_i)=s_i$.
  - s_i is an « end » event : if $\mathrm{corr}(a_{i-1})$ is a « fuse » vertex then create a diagonal between s_i and $\mathrm{corr}(a_{i-1})$. Erase a_{i-1} from T.
  - s_i is a «separating » vertex
    Search for the edge a_j located below s_i in T
    Create a diagonal between s_i and $\mathrm{corr}(a_j)$
    Update $\mathrm{corr}(a_j)=s_i$
    Inserte a_i into T and set $\mathrm{corr}(a_i)=s_i$.

Polygon Triangulation

```
▪ $s_i$ is a « fuse » vertex
  if $\mathrm{corr}(a_{i-1})$ is a fuse vertex
    create the diagonal between $s_i$ and $\mathrm{corr}(a_{i-1})$
  Delete $a_{i-1}$ from $T$
  Search the edge $a_j$ directly below $s_i$ in $T$
  If $\mathrm{corr}(a_j)$ is a fuse vertex
    create a diagonal between $s_i$ and $\mathrm{corr}(a_j)$
  Update $\mathrm{corr}(a_j)=s_i$.

▪ $s_i$ is a « regular » vertex
    If the interior of $P$ is above $s_i$
      If $\mathrm{corr}(a_{i-1})$ is a fusion vertex
         create a diagonal between $s_i$ and $\mathrm{corr}(a_{i-1})$
      delete $a_{i-1}$ from $T$
      Insert $a_i$ into $T$ and update $\mathrm{corr}(a_i)=s_i$
    Else search in $T$ which $a_j$ is below $s_i$
      If $\mathrm{corr}(a_j)$ is a « fuse » vertex
        create a diagonal between $s_i$ and $\mathrm{corr}(a_j)$
      Update $\mathrm{corr}(a_j) = s_i$
```

Polygon Triangulation

- Performances :
  - Building the event queue : $O(n\log n)$
  - Every event takes at most $O(\log n)$, and there are exactly n events.
  - All in all, $O(n\log n)$
  - Storage : $O(n)$

Polygon Triangulation

- Triangulation of monotone polygons : again by planar sweeping
- Use a stack S, in a lexicographic order
  - Contains the vertices that have been already met but should be bound to other vertices – takes the shape of a “cone”.
  - Two cases :
    - The next vertex is on the same side of the 1st vertex on the stack
    - Or not ...

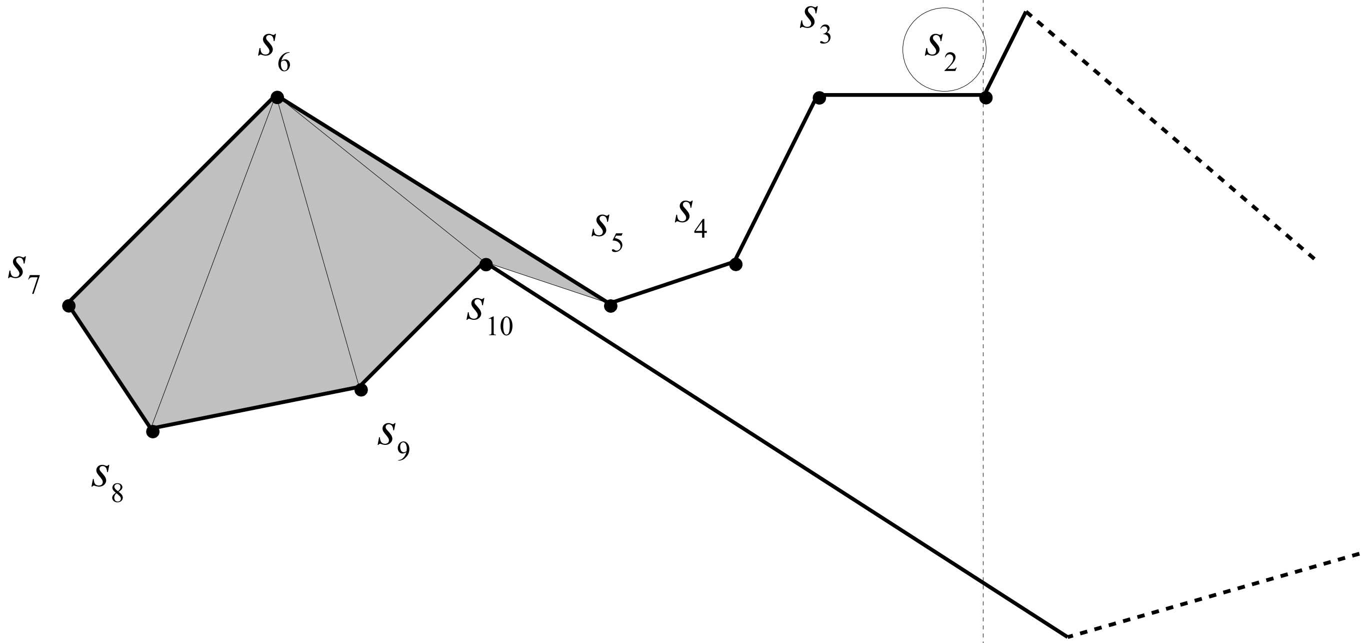
Stack S

s_3

s_4

s_5

s_{10}

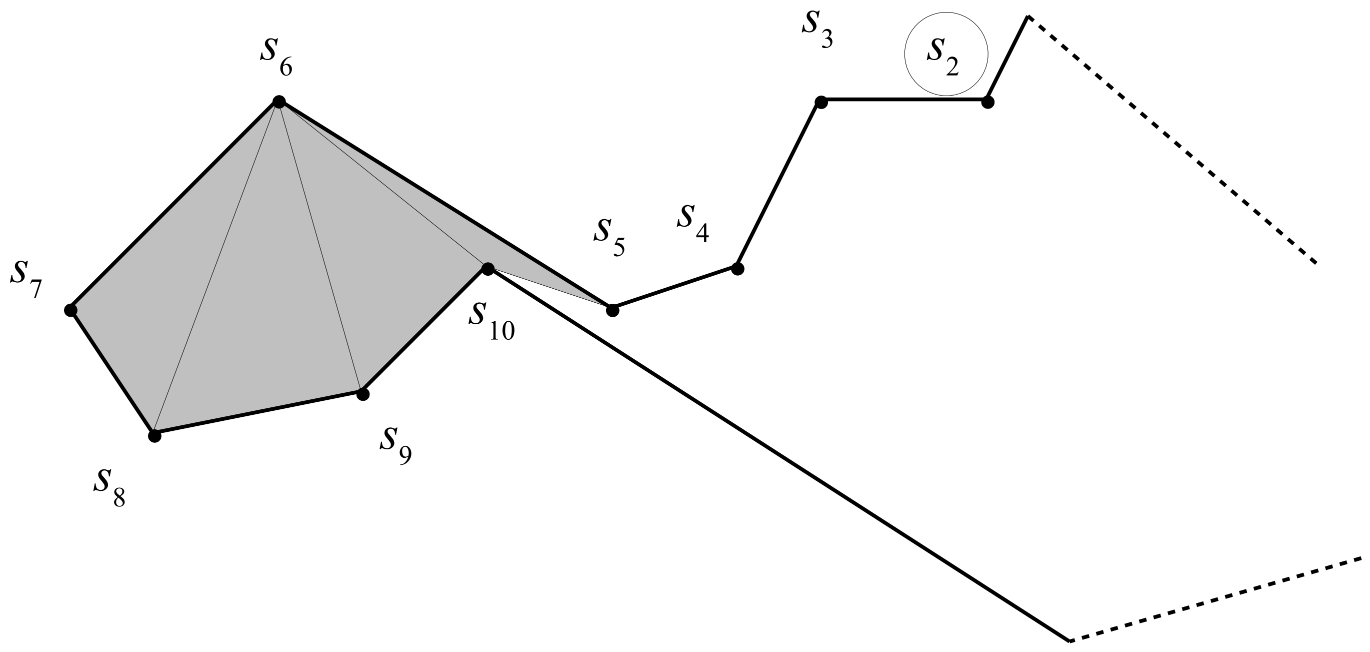


Polygon Triangulation

- If the next vertex is on the same side as the first on stack
  - One can try to link successively the vertices on the top of the stack to the current vertex.
  - At one point, we must stop (a diagonal cuts the polygon)

Stack S

s_3
s_4
s_5
s_{10}



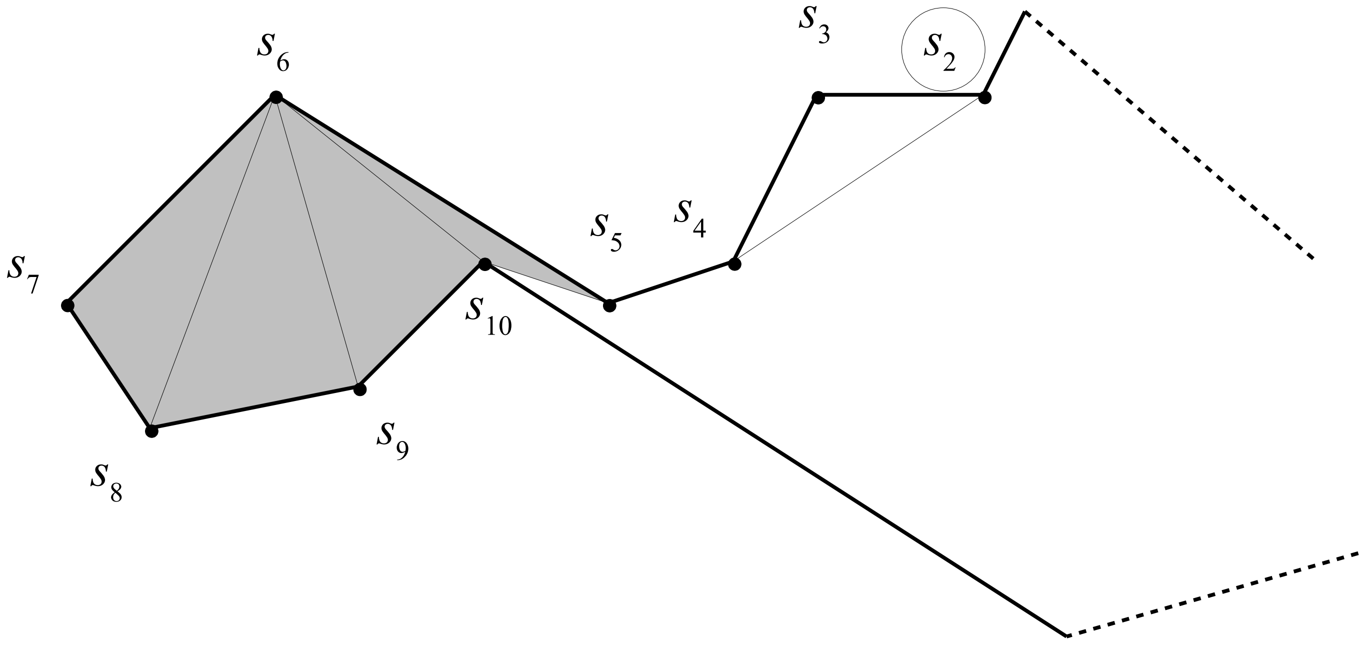
Polygon Triangulation

- If the next vertex is on the same side as the first on stack
  - One can try to link successively the vertices on the top of the stack to the current vertex.
  - At one point, we must stop (a diagonal cuts the polygon)

Stack S

s_5

s_{10}



CAD & Computational Geometry

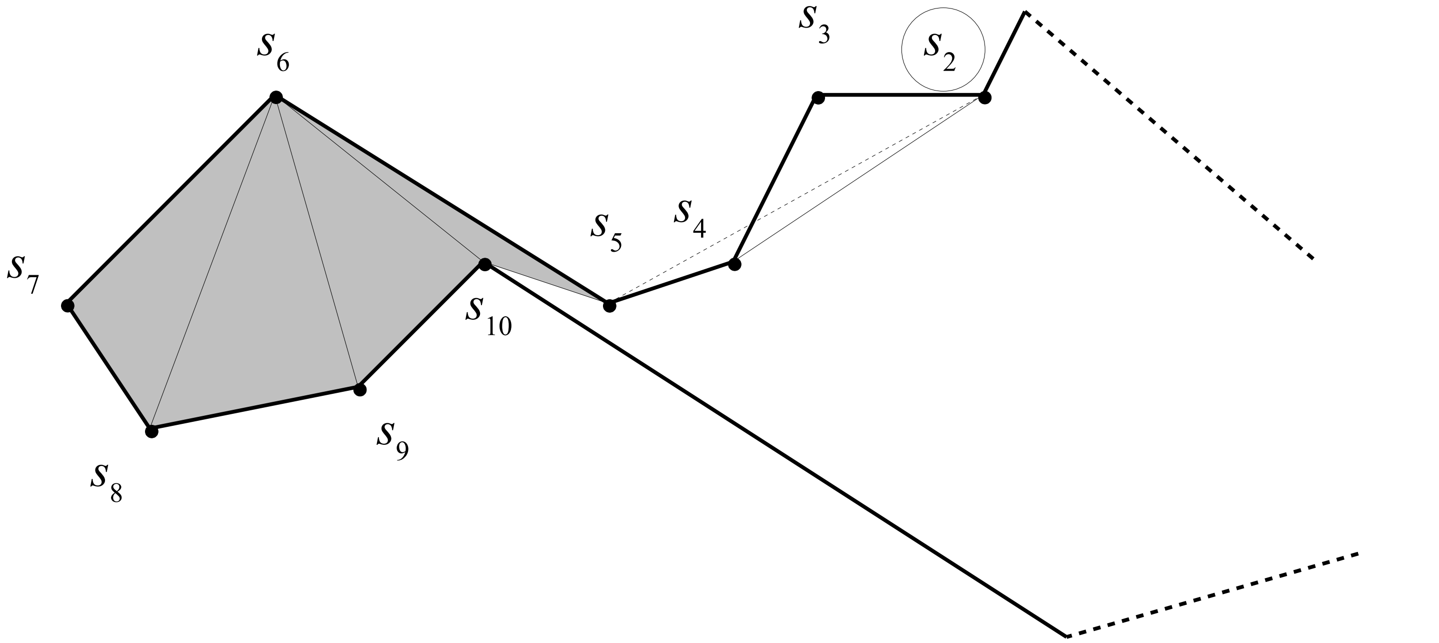
Polygon Triangulation

- If the next vertex is on the same side as the first on stack
  - One can try to link successively the vertices on the top of the stack to the current vertex.
  - At one point, we must stip (a diagonal cuts the polygon)

Stack S

s_5

s_{10}



CAD & Computational Geometry

Polygon Triangulation

- If the next vertex is on the same side as the first on stack
  - One can try to link successively the vertices on the top of the stack to the current vertex.
  - At one point, we must stip (a diagonal cuts the polygon)

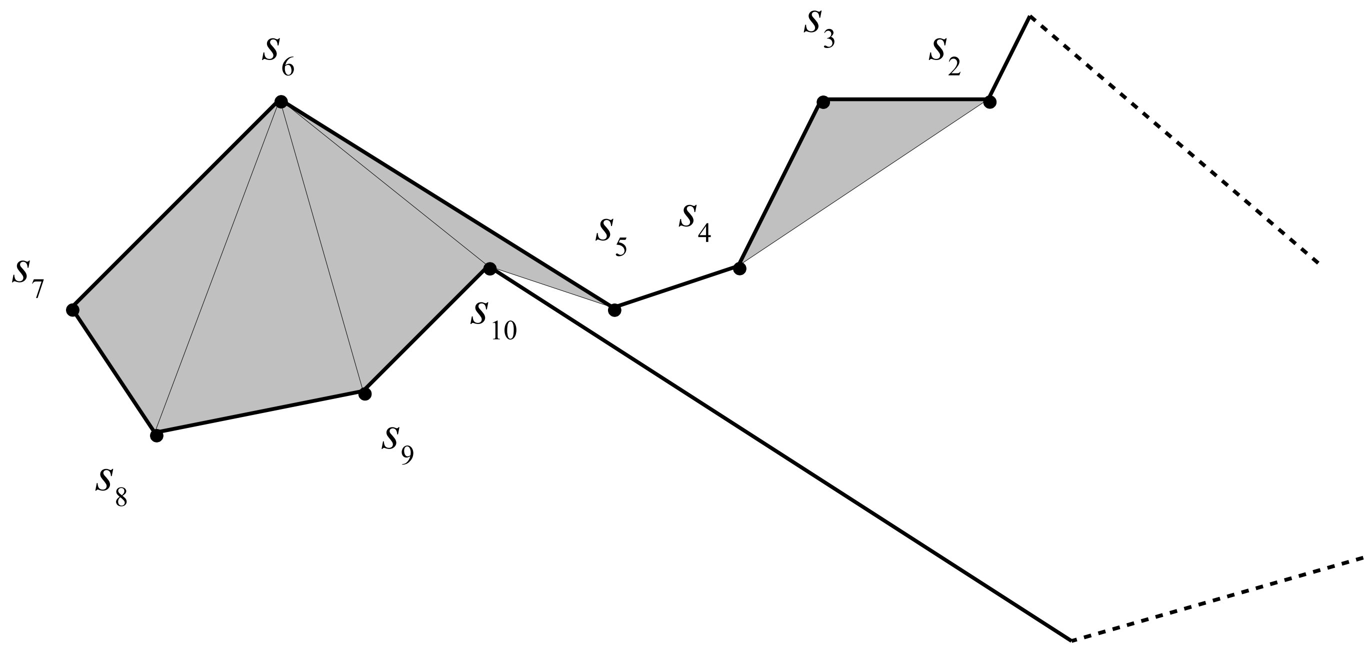
Stack S

s_2

s_4

s_5

s_{10}



  - At the end, one must push the first and last vertex linked back onto the stack

CAD & Computational Geometry

Polygon Triangulation

- If the next vertex is on the same side as the first on stack
  - One can try to link successively the vertices on the top of the stack to the current vertex.
  - Sometimes, merely push the current vertex on the stack ...

Stack S

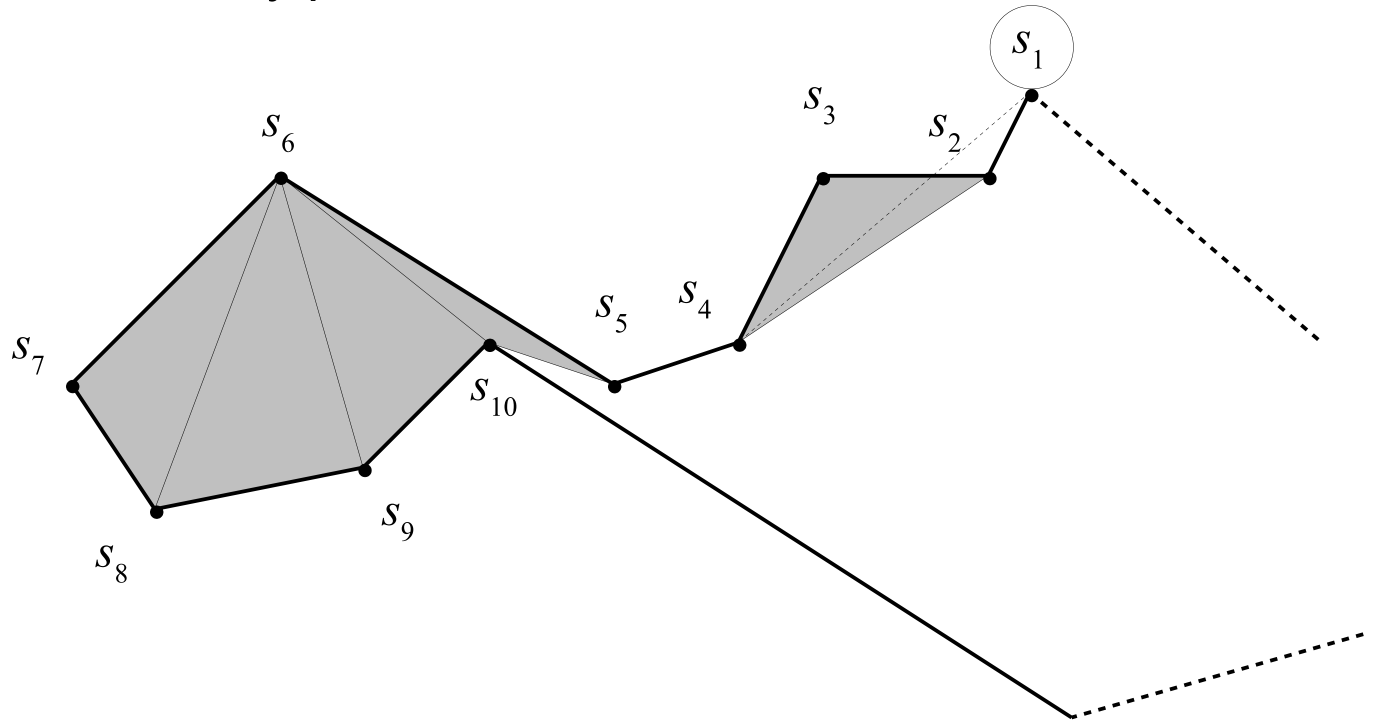
s_1

s_2

s_4

s_5

s_{10}



CAD & Computational Geometry

Polygon Triangulation

- If the next vertex is NOT on the same side as the first on stack

Stack S

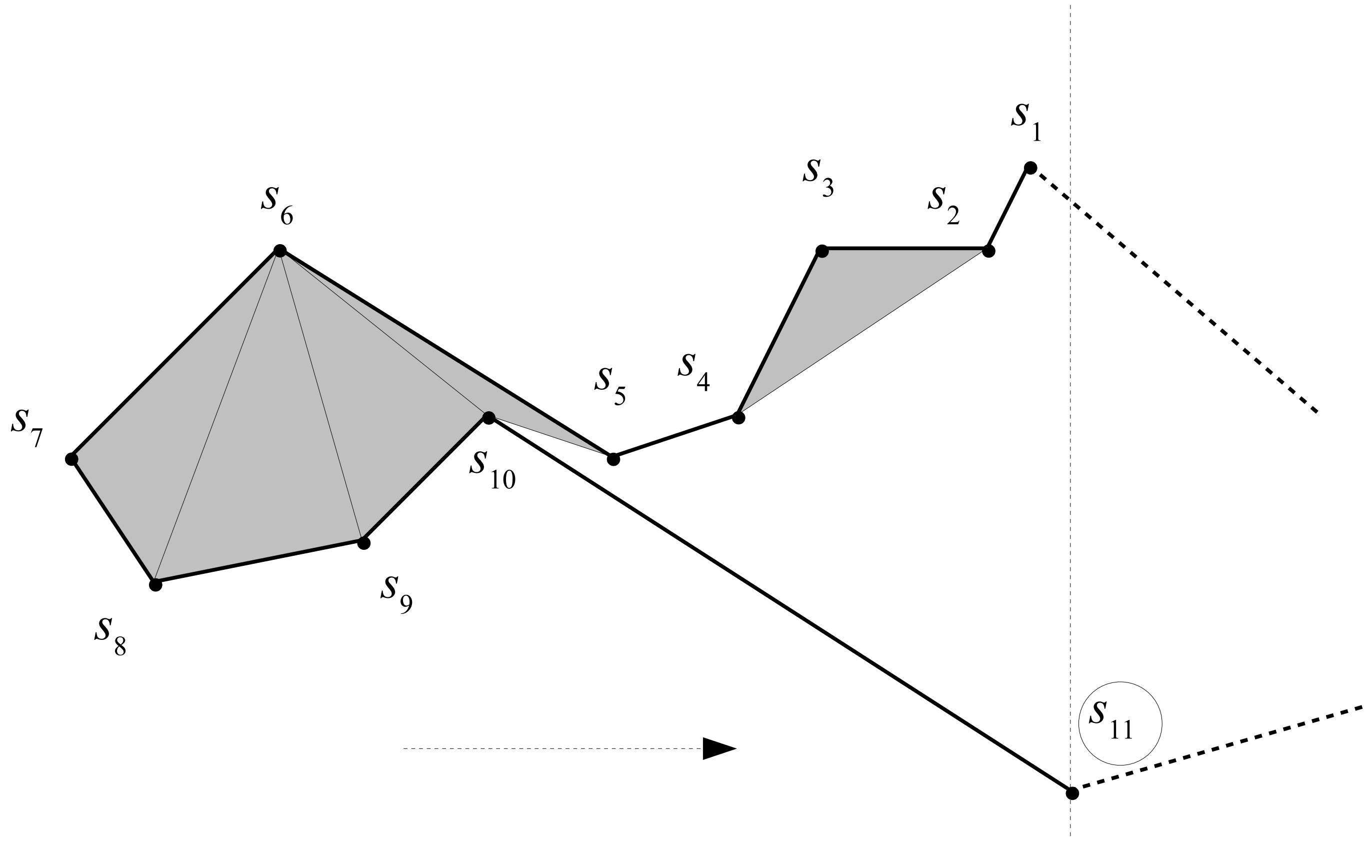
s_1

s_2

s_4

s_5

s_{10}



CAD & Computational Geometry

Polygon Triangulation

- If the next vertex is NOT on the same side as the first on stack
  - One can successively link all vertices from the stack but the last one to the current vertex

Stack S

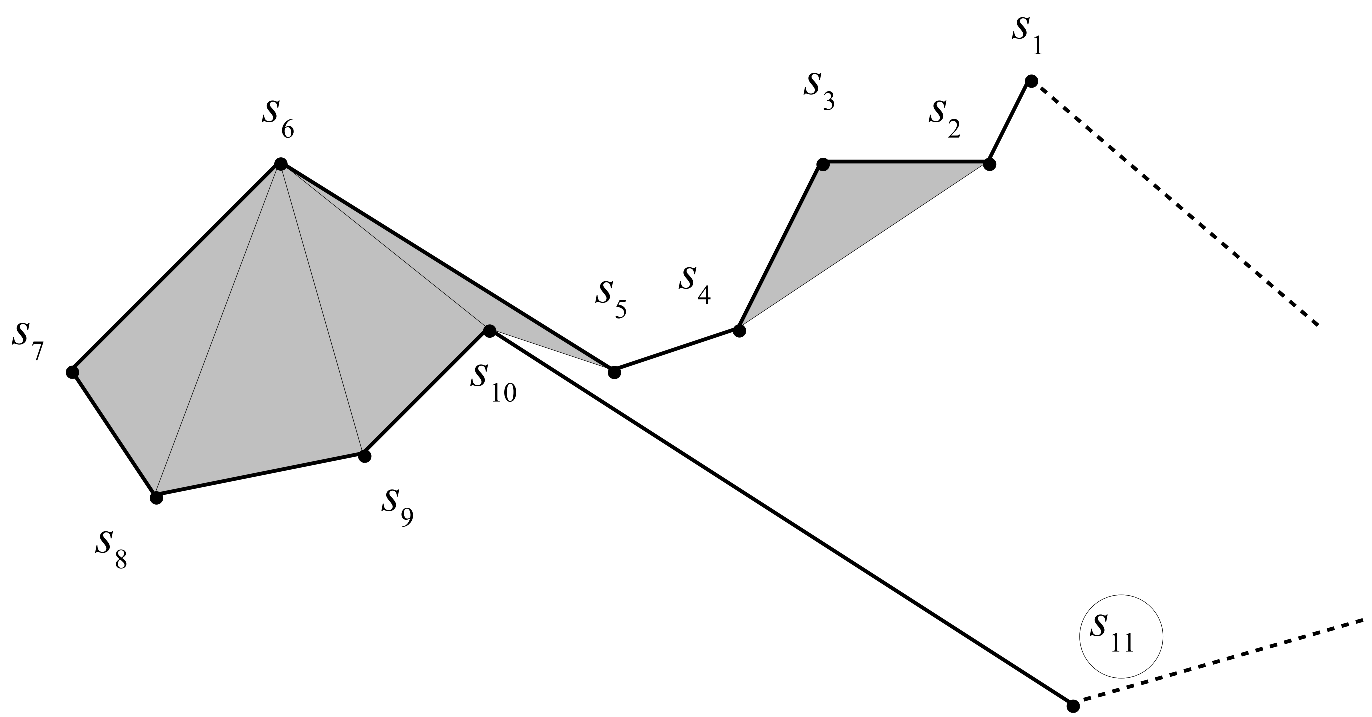
s_1

s_2

s_4

s_5

s_{10}



CAD & Computational Geometry

Polygon Triangulation

- If the next vertex is NOT on the same side as the first on stack
  - One can successively link all vertices from the stack but the last one to the current vertex

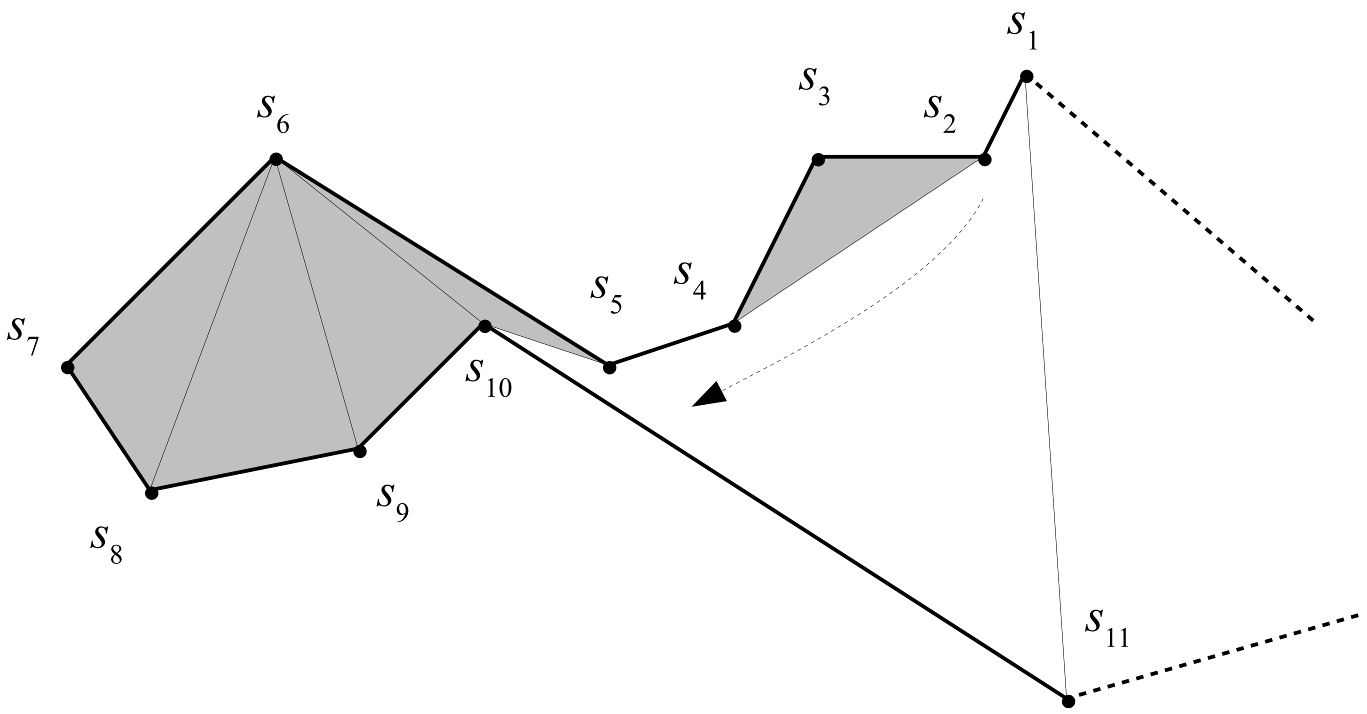
Stack S

s_2

s_4

s_5

s_{10}



Polygon Triangulation

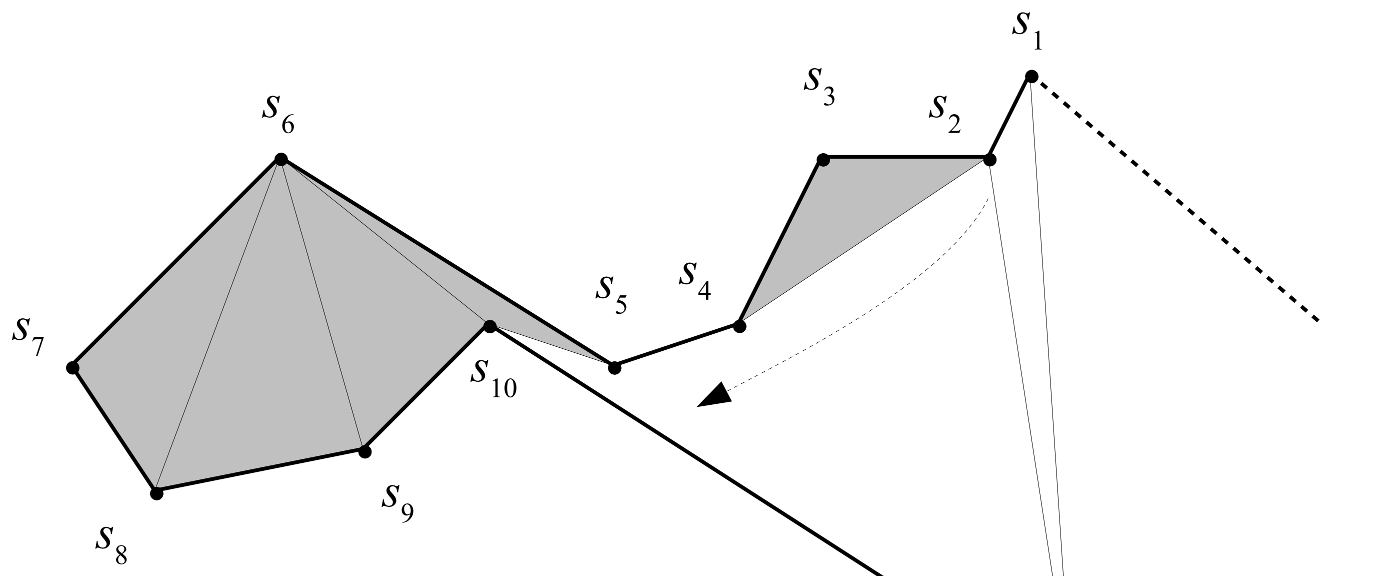
- If the next vertex is NOT on the same side as the first on stack
  - One can successively link all vertices from the stack but the last one to the current vertex

Stack S

s_4

s_5

s_{10}



CAD & Computational Geometry

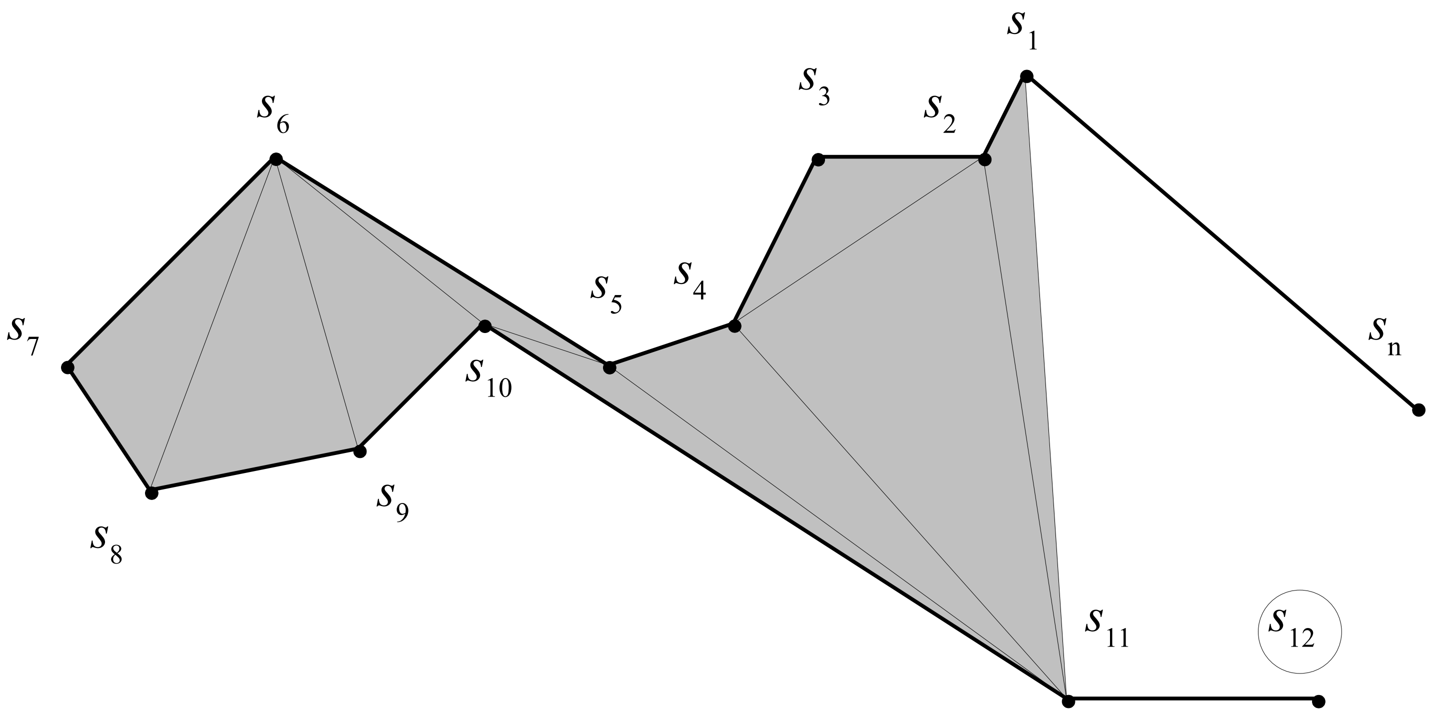
Polygon Triangulation

- If the next vertex is NOT on the same side as the first on stack
  - One can successively link all vertices from the stack but the last one to the current vertex

Stack S

s_{11}

s_1



  - At the end, the ex-first vertex is pushed on the stack again
  - One then processes the next vertex, here s_{12}

Polygon Triangulation

```
TriangulateMonotonePolygon(MP)
Input : an x-montone polygon MP
Output : a triangulation of MP
{
  Fuse the vertices of the superior and inferior chain into a common structure. The vertices are classified in a
lexicographic order, $t_0$ is the first (leftmost) and, $t_{n-1}$ the last one.
  Initialize an empty stack S , and push $t_0$ and $t_1$
  For  j from 2 to n-2
  {
    If $t_j$ and the 1st vertex on the stack are on different chains
    {
      Pop all the vertices from the stack, create a diagonal between those and the current vertex $t_j$,
        except the last on the stack.
      Push $t_{j-1}$ and $t_j$ on the stack
    }
    Else
    {
      Pop one vertex from the stack S ;
      Pop the other vertices one by one as far as one can make a diagonal (not crossings)
      Push back the last pop-ed vertex, and $t_j$
    }
  }
  Create diagonals between $t_{n-1}$ and all the remaining vertices in the stack, except the first and last.
}
```

Polygon Triangulation

- Complexity of the algorithm
  - Ordering of vertices is in $O(n)$ because the polygon is monotone !
  - The loop is done n-3 times
    - In the worst case, $2n$-4 vertices are “pushed” on the stack
    - The number of “popped” vertices cannot exceed this value $2n$-4
    - Each individual operation is in constant time $O(1)$
- The complexity is therefore $O(n)$

Polygon Triangulation

- Complexity of the complete triangulation algorithm
  - Decomposition in monotone polygons : $O(nlogn)$
  - It generates k monotone polygons, each van be triangulated in $O(n_i)$. Of course, $\Sigma n_i = O(n)$
    - Therefore the whole thing is in $O(n)$
- As a consequence, the total complexity is dominated by the decomposition into monotone polygons : $O(nlogn)$

Polygon Triangulation

- Some remarks
  - It is not an optimal algorithm for simple polygons (without holes)
    - There exist more involved alternatives in $O(n)$
      - B. Chazelle. Triangulating a simple polygon in linear time. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, IEEE, 1990, pages 220-230.
    - But it is more general because it works with holes. In this case, it is actually optimal, since the theoretical limit is precisely $\Omega(n \log n)$.

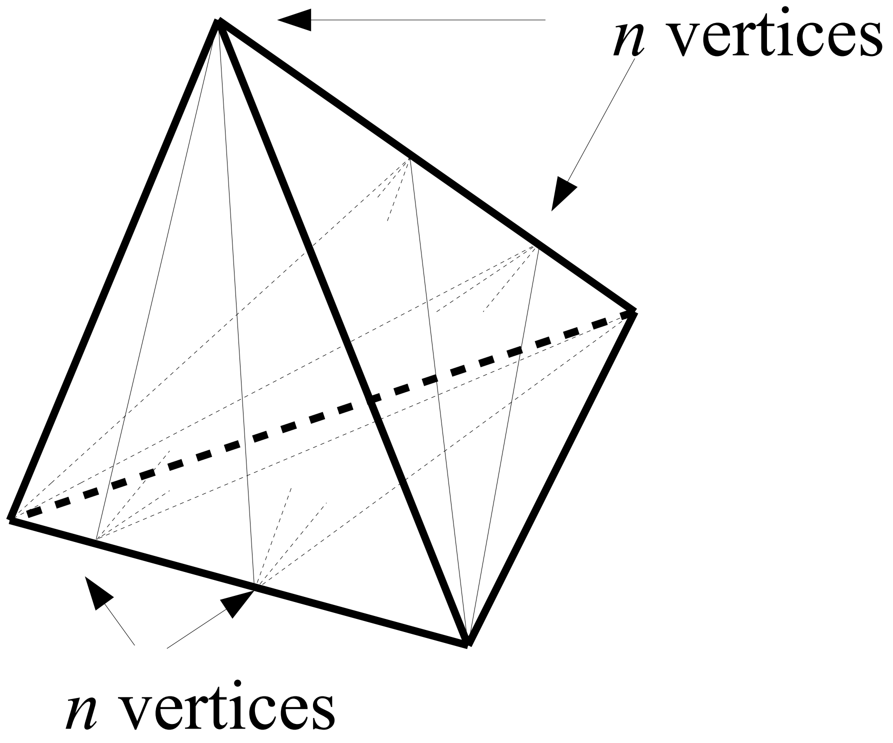
Polygon Triangulation

- In 3D : “tetrahedralization” of an n-sided polytope
  - Much, much more difficult !
  - Sometimes impossible without adding internal vertices (called Steiner points)
  - Knowing if one needs Steiner points : NP – hard.

    J Ruppert, R Seidel, On the difficulty of triangulating three-dimensional nonconvex polyhedra, *Discrete & Computational Geometry*, 1992, Springer.
  - The number of steiner points is bound by $\Theta(n+r^2)$, r is the numbler of reflex edges (for a non convex polytope)
  - The number of tetrahedrons is not linear with the number of vertices on the boundary of the polytope...

Polygon Triangulation

- Simple polyhedron that does not yield a triangulation in $O(n)$ tetrahedrons...

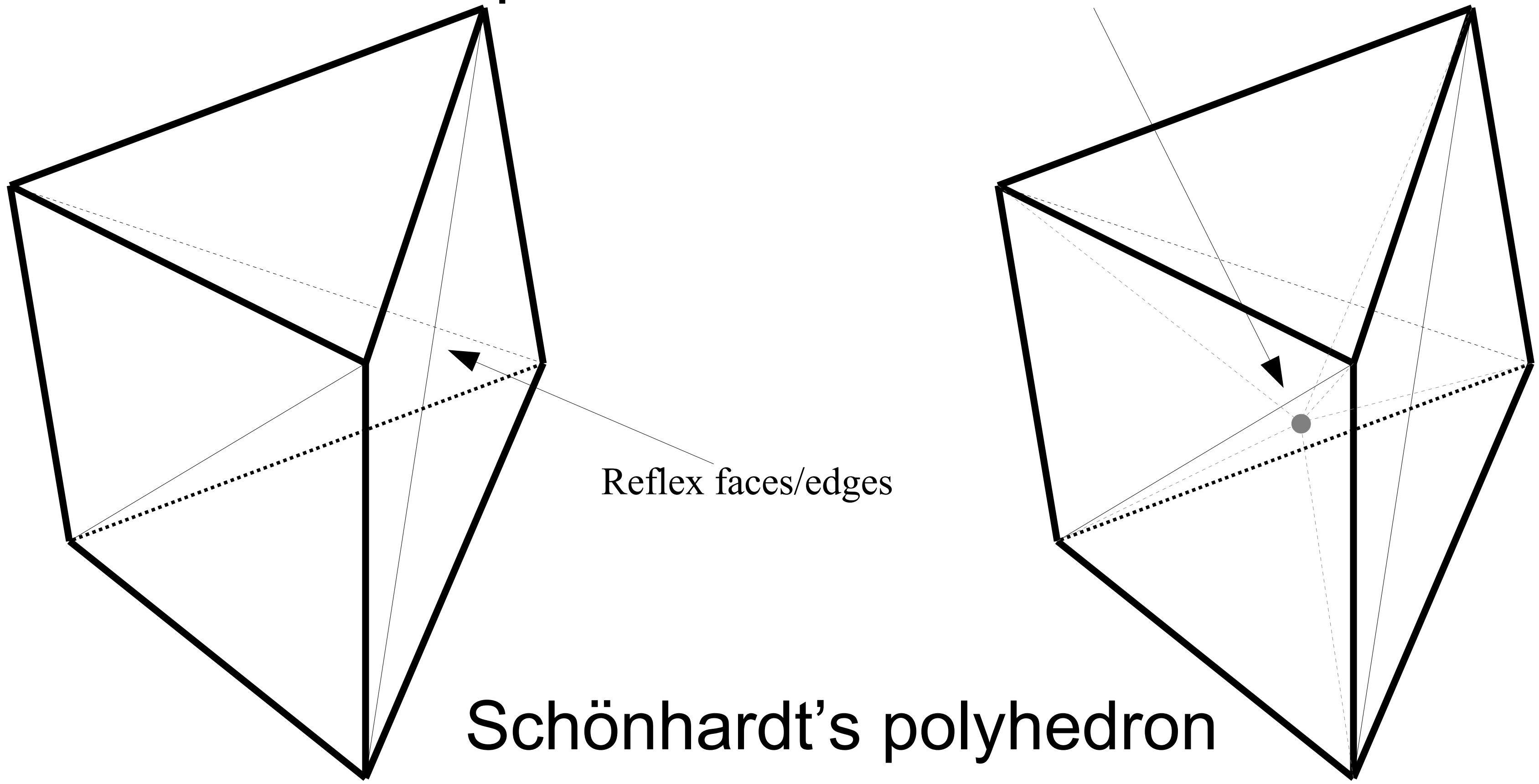


Totally, there are $2n$ vertices and $(n-1)^2$ tetrahedrons

CAD & Computational Geometry

Polygon Triangulation

- Example of simple polyhedron that is not tessellable without Steiner points…



Schönhardt’s polyhedron

E. Schönhardt. Über die Zerlegung von Dreieckspolyedern in Tetraeder. *Math. Ann.*, 98:309-312,1928.